Министерство образования и науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ
ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Муромцев Д.И., Колчин М.А.

Разработка экспертных систем в Drools Guvnor

Учебное пособие

Санкт-Петербург
2013

УДК [004.891 + 002.53:004.89] (075.8)

Д.И. Муромцев, М.А. Колчин. Разработка экспертных систем в Drools Guvnor – СПб: НИУ ИТМО, 2013. – 54 с.

В учебном пособии представлены лабораторные работы, позволяющие студентам овладеть основными навыками разработки экспертных систем с помощью Drools Guvnor. Рассматриваются основные аспекты разработки баз знаний, продукционная и табличная формы представления знаний, а также принципы формирования запросов к базам знаний.

Методическое пособие адресовано студентам высших учебных заведений, обучающихся по направлению 210202.65.08 «Проектирование и технология электронных средств» и по специальности 0900104.65 «Комплексная защита объектов информатизации».

Одобрено на заседании совета факультета компьютерных технологий и управления Санкт-Петербургского государственного университета информационных технологий механики и оптики, протокол № 1 от 31 января 2013 года.

В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России, которым присвоена категория «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена программа его развития на 2009–2018 годы. В 2011 году Университет получил наименование «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики».

Оглавление

Введение

Система управления бизнес-правилами (Business Rule Management System) – это информационная система, которая используется для ведения, поддержки и исполнения логики принятия решений, используемой информационными системами внутри организации или предприятия.

Эта логика, чаще называемая *бизнес-правилами*, описывает различные политики, требования и ограничения, которые используются для принятия решений, имеющих место в информационных системах.

Система управления бизнес-правилами состоит из следующих *основных компонентов*:

- репозиторий, который хранит базы знаний, в которые объединяются бизнес-правила, и позволяет отделить логику принятия решений от программного кода системы;
- инструменты, позволяющие как разработчикам, так и экспертам определять и поддерживать бизнес-правила;
- среда выполнения, позволяющая приложениям управлять бизнес-правилами, разработанными в BRMS и выполнять их, используя машину вывода;

Бизнес-правила по своей сути – это развитие продукционных правил и систем в сфере бизнеса и копроративных информационных систем.

В первой части пособия приведены основные теоретические вопросы построения экспертных систем. Подробнее с теорией и технологией экспертных систем вы можете познакомиться в учебном пособии Д.И. Муромцева Введение в технологию экспертных систем - СПб. СПбГУ ИТМО, 2005.

В данном пособии рассматривается разработка бизнес-правил в системе Drools Guvnor[1]. В качестве примера будет разработана небольшая база знаний для системы принятия решений о выдачи потребительского кредита.

[1] Drools Guvnor - http://www.jboss.org/drools/

Архитектура и этапы разработки ЭС

Создаваемая экспертная система проектирования объективов (ЭСПО) является программным продуктом, позволяющим помочь пользователю при поиске стартовой точки оптической системы, что пока не реализовано ни одной из существующих программ. Очевидно, что новое программное обеспечение (ПО) должно непосредственно стыковаться с наиболее распространенным специализированным ПО, например, CODEV [Synopsys, 2012], OSLO [OSLO, 2012], SYNOPSYS [Synopsys, 2012], ZEMAX [Radiant Zemax, 2012] и др. Следует отметить, что перечисленные программы предоставляют пользователям некоторый ограниченный набор возможностей поиска стартовых точек, например, каталоги готовой продукции [Edmund scientific, 2012], патентная литература [Europatent, 2012]. Попытка создания экспертной системы была предпринята только Доном Дилворсом [Dilworth, 1987].

В случае построения экспертной системы для проектирования объективов нами решается задача выбора исходной оптической схемы объектива (стартовой точки) в виде структурной схемы, представленной как последовательность оптических элементов с указанием их типа и взаимного расположения. Условимся называть эту последовательность «формулой структурного синтеза объектива».

Задача решена, если получены формулы структурного синтеза, пригодные для дальнейших манипуляций с ними: определения параметров объектива (параметрический синтез), введения найденных параметров в специализированную программу проектирования оптических систем, результатом работы которой и будет оптическая схема объектива, удовлетворяющая требованиям технического задания.

Структура и принцип работы ЭС

В настоящей работе в качестве рабочего определения экспертной системы примем вариант, приведенный в трудах Т.А. Гавриловой [Гаврилова, 2001], который, по нашему мнению, наиболее полно отражает смысл функционирования экспертных систем.

> *Экспертные системы (ЭС) — это сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных областях и тиражирующие этот эмпирический опыт для консультаций менее квалифицированных пользователей.*

В целом процесс функционирования ЭС можно представить следующим образом: пользователь, желающий получить необходимую информацию, через пользовательский интерфейс посылает запрос к ЭС; решатель, пользуясь базой знаний, генерирует и выдает пользователю

необходимую рекомендацию, объясняя ход своих рассуждений при помощи подсистемы объяснений.

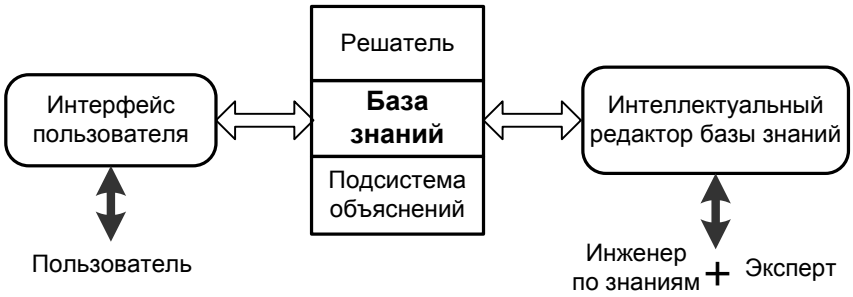

Рисунок 1 - Структура экспертной системы

Обобщенная структура экспертной системы представлена на рис. 1. Следует учесть, что существующие ЭС могут иметь более сложную структуру, однако приведенные на рисунке элементы — неотъемлемая часть любой ЭС.

Определим функции и роли основных элементов ЭС.

Интерфейс пользователя — комплекс программ, реализующих диалог конечного пользователя с ЭС как на стадии ввода информации, так и при получении результатов.

База знаний (БЗ) — ядро ЭС, совокупность знаний предметной области, записанная на машинный носитель в форме, понятной эксперту и пользователю (обычно на некотором языке, приближенном к естественному). Параллельно такому «человеческому» представлению существует БЗ во внутреннем «машинном» представлении.

Решатель (дедуктивная машина, машина вывода, блок логического вывода) — программа, моделирующая ход рассуждений эксперта на основании информации, имеющейся в БЗ.

Подсистема объяснений — программа, позволяющая пользователю получить ответы на вопросы: «Как была получена та или иная рекомендация?», т.е. описание всего процесса получения решения с указанием использованных фрагментов БЗ (всех шагов цепи умозаключений), «Почему система приняла такое решение?», т.е. ссылка на умозаключение, непосредственно предшествовавшее полученному решению — отход на один шаг назад. Развитые подсистемы объяснений поддерживают и другие типы вопросов.

Интеллектуальный редактор БЗ — программа, предоставляющая инженеру по знаниям возможность создавать БЗ в диалоговом режиме. Включает в себя систему вложенных меню, шаблонов языка представления знаний, подсказок («help»-режим) и других сервисных средств, облегчающих работу с базой. Данный элемент ЭС предназначен для работы инженера по знаниям (когнитолога, инженера-интерпретатора,

аналитика) — специалиста по искусственному интеллекту, выступающего в роли промежуточного звена между экспертом и базой знаний.

Еще раз следует подчеркнуть, что представленная на рис. 1 структура содержит лишь необходимый минимум, что означает обязательное присутствие указанных на ней элементов; если система представлена разработчиками как экспертная, это гарантирует наличие аппарата обработки знаний. Однако промышленные прикладные ЭС могут быть существенно сложнее и дополнительно включать базы данных, интерфейсы обмена данными с различными пакетами прикладных программ, электронными библиотеками и т.д.

Классификация ЭС

В общем случае все экспертные системы, основанные на знаниях, можно подразделить на системы, решающие задачи анализа (интерпретация данных, диагностика, поддержка принятия решения) и решающие задачи синтеза (проектирование, планирование, управление). Основное отличие задач анализа от задач синтеза заключается в том, что если в первых все элементы множества решений могут быть описаны и включены в систему, то в задачах синтеза множество решений потенциально не ограничено и строится путем комбинирования компонентов или подпроблем. Некоторые задачи (обучение, мониторинг, прогнозирование) сочетают в себе как анализ, так и синтез решения.

Интерпретация данных — одна из традиционных задач для экспертных систем. Под интерпретацией понимается процесс определения семантики данных, результаты которого должны быть непротиворечивыми (согласованными) и корректными, обычно предусматривается многовариантный анализ данных. Приведем примеры систем интерпретации данных [Попов и др. (ред.), 1990; Хейес-Рот и др., 1987; Джарратано, Райли, 2006]:

- SIAP — обнаружение и идентификация различных типов океанских судов по результатам аэрокосмического сканирования;
- АВТАНТЕСТ, МИКРОЛЮШЕР и др. — определение основных свойств личности по результатам психодиагностического тестирования в системах.

Диагностика — процесс соотнесения объекта с некоторым классом объектов или обнаружение неисправности в некоторой системе. Неисправность — это отклонение от нормы. Такая трактовка позволяет с единых теоретических позиций рассматривать и неисправность оборудования в технических системах, и заболевания живых организмов, и всевозможные природные аномалии. Специфика процесса требует явного описания функциональной структуры («анатомии») диагностирующей системы. Примеры:

- ANGY — диагностика и терапия сужения коронарных сосудов;

- MANAGE — диагностика заболеваний риса;
- CRIB — диагностика ошибок в аппаратуре и математическом обеспечении ЭВМ.

Мониторинг — непрерывная интерпретация данных в реальном масштабе времени и предупреждение о выходе тех или иных параметров за допустимые пределы. Главные проблемы, возникающие при функционировании систем мониторинга — это возможные «пропуски» критических ситуаций, а также «ложные» срабатывания в штатных ситуациях. Сложность преодоления этих проблем — в размытости симптомов тревожных ситуаций и необходимости учета временного контекста. Примеры:

- REACTOR — контроль работы электростанций СПРИНТ, помощь диспетчерам атомного реактора;
- FALCON — контроль аварийных датчиков на химическом заводе.

Проектирование заключается в подготовке спецификаций на создание «объектов» с заранее определенными свойствами. Под спецификацией понимается весь набор необходимых документов — чертеж, пояснительная записка и т.д. Для организации эффективного проектирования, и в еще большей степени — перепроектирования, необходимо формировать не только сами проектные решения, но и вырабатывать мотивы их принятия. Таким образом, в задачах проектирования тесно связаны два основных процесса, выполняемых в рамках соответствующей ЭС — вывод решения и объяснение. Примеры:

- XCON (или R1) — проектирование конфигураций ЭВМ VAX-11/780;
- CADHELP — проектирование БИС;
- SYN — синтез электрических цепей.

Прогнозирование позволяет предсказывать последствия некоторых событий или явлений на основании анализа имеющихся данных. Прогнозирующие системы формируют описания вероятных последствий заданных ситуаций. В прогнозирующей системе обычно используется параметрическая динамическая модель, в которой значения параметров «подгоняются» под заданную ситуацию. Получаемые на основе этой модели оценки составляют базу для прогнозов с вероятностными оценками. Примеры:

- WILLARD — предсказание погоды;
- PLANT — оценка будущего урожая;
- ECON и др. — прогнозы в экономике.

Планирование — составление планов действий, относящихся к объектам, способным выполнять некоторые функции. В таких ЭС используются модели поведения реальных объектов, с тем чтобы логически вывести последствия планируемой деятельности. Примеры:

- STRIPS — планирование поведения робота;
- ISIS — планирование промышленных заказов;

- MOLGEN — планирование химического эксперимента.

Обучение — построение автоматизированной обучающей системы, частично выполняющей функции преподавателя. Обучающие системы с помощью ЭВМ выявляют ошибки при изучении какой-либо дисциплины и предлагают правильные решения. Они аккумулируют знания о «виртуальном» ученике и его характерных ошибках, способны выявить проблемы в познаниях обучаемых и находить соответствующие средства для их ликвидации. Кроме того, в зависимости от успешности решения учеником заданий они формируют план занятий с ним. Примеры:

- Учитель LISP — обучение языку программирования LISP;
- PROUST — обучение языку Паскаль и др.

Управление — функция организованной системы, поддерживающая определенный режим деятельности. Такого рода ЭС осуществляют управление поведением сложных систем в соответствии с заданными спецификациями. Примеры:

- GAS — управление газовой котельной;
- CALEX — управление сельскохозяйственным комплексом;
- Project Assistant — управление системой календарного планирования.

Поддержка принятия решений — это совокупность процедур, обеспечивающих лицо, принимающее решение, необходимой информацией и рекомендациями. ЭС помогают специалистам выбрать из множества и/или сформировать нужный вариант действий при принятии ответственных решений. Примеры:

- CRYSIS — выбор стратегии выхода фирмы из кризисной ситуации;
- LIMEX — советы по борьбе с сельскохозяйственными вредителями;
- CHOICE — помощь в выборе страховой компании или инвестора.

Классификация по связи с реальным временем

Общая классификация ЭС по связи с реальным временем приведена на рис. 2.

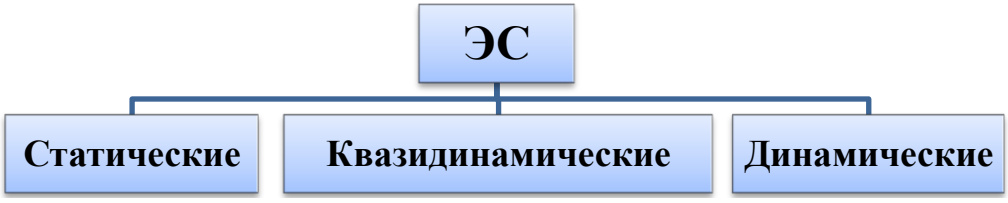

Рисунок 2 - Классификация экспертных систем по связи с реальным временем

Статические ЭС разрабатываются для предметных областей, в которых база знаний и интерпретируемые данные не изменяются во времени (например, диагностика неисправностей в автомобиле).

Квазидинамические ЭС интерпретируют ситуацию, которая изменяется с некоторым фиксированным интервалом времени. Например, ЭС анализа лабораторных измерений, для которых исходные данные поступают один раз в несколько часов, выполняют анализ динамики полученных показателей по отношению к предыдущему измерению.

Динамические ЭС работают в режиме реального времени в сопряжении с датчиками объектов. Эти системы выполняют непрерывную интерпретацию поступающих данных. К примеру, управление гибкими производственными комплексами, мониторинг в реанимационных палатах и т.д., а также программный комплекс для разработки динамических систем G2.

Классификация по платформе

Экспертные системы могут предназначаться для работы на компьютерах общего назначения и на специализированных вычислительных машинах (рис. 3).

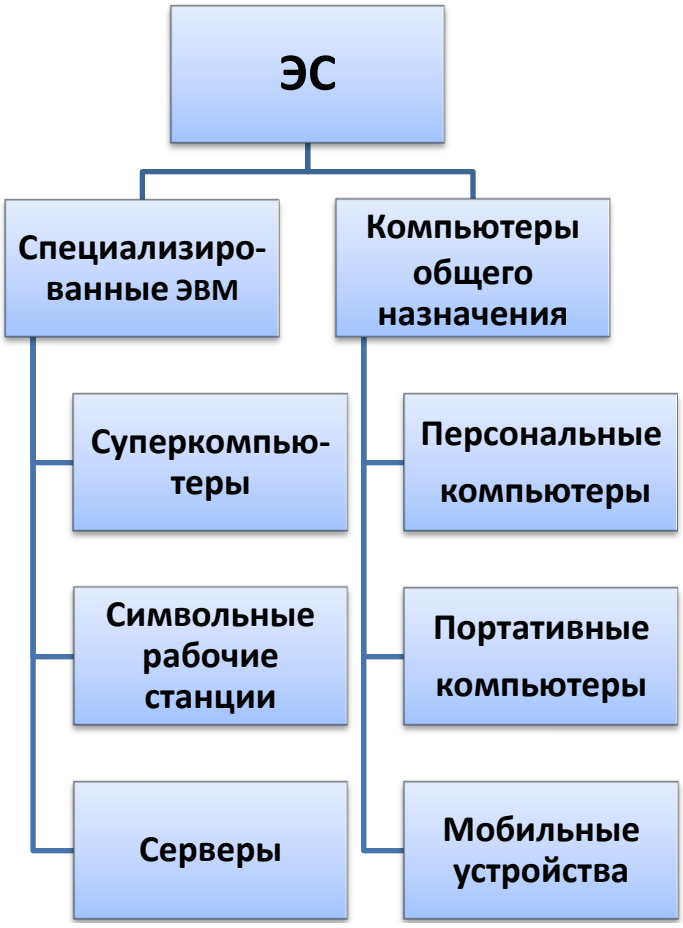

Рисунок 3 - Классификация экспертных систем по платформе

Уникальные по своим характеристикам, например системы реального времени, обрабатывающие большие массивы данных, а также стратегически важные задачи, требуют использования в качестве вычислительной платформы суперкомпьютеров (Эльбрус, CRAY, CONVEX и др.) или специализированных символьных процессоров и рабочих станций (SUN, Silicon Graphics, APOLLO). Однако большинство ЭС создаются для работы на персональных компьютерах общего назначения и даже мобильных устройствах.

Классификация по степени интеграции с другими программами

Классификация ЭС по степени интеграции с другим программным обеспечением представлена на рис. 4.

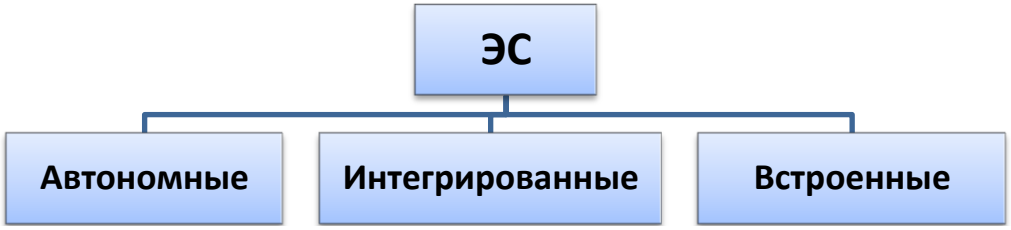

Рисунок 4 - Классификация экспертных систем по интеграции

Автономные ЭС работают непосредственно в режиме диалога с пользователем для решения типичных «экспертных» задач, когда не требуется привлекать традиционные методы обработки данных (расчеты, моделирование и т.д.).

Гибридные ЭС представляют собой программный комплекс, интегрирующий пакеты прикладных программ (например, математическую статистику, линейное программирование или системы управления базами данных) и инструменты для обработки знаний. Это может быть интеллектуальная надстройка над математическими пакетами или интегрированная среда для решения сложной задачи с элементами экспертных знаний.

Следует отметить, что, несмотря на внешнюю привлекательность гибридного подхода, разработка таких систем являет собой задачу на порядок более сложную, чем разработка автономной ЭС. Стыковка не просто разных пакетов, а разных методологий (что свойственно гибридным системам) порождает целый комплекс теоретических и практических трудностей.

Коллектив разработчиков экспертной системы

В целом за разработку экспертных систем целесообразно браться организации, накопившей опыт в автоматизации таких рутинных процедур обработки информации, как:

- формирование корпоративных информационных систем;

- организация сложных расчетов;
- работа с компьютерной графикой;
- обработка текстов и автоматизированный документооборот.

В ходе решения таких задач, во-первых, происходит подготовка высококвалифицированных специалистов по информатике для создания интеллектуальных систем, во-вторых, появляется возможность исключить из рассмотрения неэкспертные задачи.

Над созданием ЭС трудится коллектив разработчиков (КР), в состав которого входят все специалисты, ответственные за создание ЭС, а не только программисты. КР составляют, по крайней мере, три человека — пользователь, эксперт и инженер по знаниям. Таким образом, минимальный коллектив включает четыре человека; в действительности он может достигать десяти человек. Увеличение коллектива разработчиков происходит по следующим причинам: необходимость учета мнения нескольких пользователей, помощи нескольких экспертов; потребность в прикладных и системных программистах. Также в этот коллектив могут входить менеджер и технический помощник. При отсутствии профессионального менеджера руководителем группы, участвующим во всех стадиях разработки, является инженер по знаниям, поэтому к его квалификации предъявляются самые высокие требования. В целом уровень и численность группы зависят от характеристик поставленной задачи.

Рассмотрим два аспекта характеристик членов КР: А — психологический, Б — профессиональный.

Пользователь

К пользователю предъявляются самые «слабые» требования, поскольку его не выбирают. По сути, он является заказчиком системы. Желательные качества: а) дружелюбие; б) умение объяснять свои требования к системе; в) отсутствие психологического барьера к применению вычислительной техники; г) интерес ко всему новому. От пользователя зависит, будет ли применяться разработанная ЭС. Замечено, что наиболее ярко качества в) и г) проявляются в молодом возрасте, поэтому иногда такие пользователи охотнее применяют ЭС, при этом у них не возникает чувства неуверенности от того, что компьютер им что-то подсказывает.

Необходимо, чтобы пользователь имел некоторый базовый уровень квалификации, который позволит ему правильно истолковать рекомендации ЭС. Для этого необходимо обеспечить полное соответствие в терминологии интерфейса ЭС и той, которая привычна и удобна для работы пользователя. Обычно требования к квалификации пользователя не очень велики, иначе он переходит в разряд экспертов и совершенно не нуждается в ЭС.

Эксперт

Эксперт — чрезвычайно важный участник команды. Он, в конечном счете, задает уровень компетенции базы знаний. Желательные качества: а) доброжелательность; б) готовность поделиться своим опытом; в) умение объяснять (педагогические навыки); г) интерес (в том числе материальный) к успешности разработки. Эксперт обычно старше остальных членов коллектива, что необходимо учитывать всем участникам разработки. Часто встает вопрос о количестве экспертов. Поскольку проблема совмещения подчас противоречивых знаний остается открытой, обычно с каждым из экспертов работают индивидуально, иногда создавая альтернативные БЗ.

Помимо высокого уровня профессиональных знаний в предметной области желательно ознакомление эксперта с основами технологии экспертных систем для более эффективного прохождения этапа извлечения знаний.

Программист

Известно, что среди представителей разных профессий программисты обладают самой низкой потребностью в общении. Однако при разработке ЭС необходим тесный контакт членов группы, поэтому желательны следующие качества: а) коммуникабельность; б) способность отказаться от традиционных навыков и освоить новые методы; в) интерес к разработке.

Поскольку современные ЭС — сложнейшие дорогостоящие программные комплексы, программисты помимо опыта и навыков разработки программ обязательно должны ознакомиться с основными структурами представления знаний и механизмами вывода, состоянием рынка программных продуктов для разработки.

Инженер по знаниям

Спрос на аналитиков — уникальных специалистов по работе со знаниями — постоянно растет. На современном рынке информационных технологий потребность в аналитиках многократно превышает спрос на программистов. Этих специалистов называют по-разному — системные аналитики, постановщики задач, инженеры по знаниям, инженеры-когнитологи, бизнес-аналитики. Близкие понятия и синонимы: постановщик задач / инженер-когнитолог / CKO (chief knowledge officer) — директор по управлению знаниями / менеджер знаний / функциональный аналитик или бизнес-аналитик.

Существуют такие профессии и виды деятельности, для которых природные качества личности (направленность, способности, темперамент) могут иметь характер абсолютного показания или противопоказания к занятиям. По-видимому, инженерия знаний принадлежит к таким областям. По различным оценкам, это одна из самых малочисленных, высокооплачиваемых и дефицитных в мире специальностей. При определении профессиональных требований к

аналитику следует учитывать, что ему необходимы различные навыки и умения для грамотной и эффективной организации процессов извлечения, концептуализации и формализации знаний.

Успешность подбора и уровень подготовки коллектива разработчиков ЭС определяют эффективность и продолжительность всего процесса разработки.

Разработка промышленных ЭС

Разработка программных комплексов экспертных систем перестала быть искусством и перешла в область индустрии. Сравнительно медленный темп этого перехода связан с тем, что долгое время эксплуатация систем искусственного интеллекта начиналась уже на этапе проектирования, в ходе которого разрабатывалось несколько прототипных версий программ, прежде чем был получен конечный продукт. Такой подход хорош в исследовательских условиях, однако для коммерческого подхода он слишком дорог, чтобы оправдать затраты на разработку.

Процесс разработки промышленной экспертной системы, опираясь на традиционные технологии [Николов и др., 1991; Waterman, 2004], можно разделить на шесть более или менее независимых этапов (рис. 5), практически не связанных с предметной областью.

Рисунок 5 - Этапы разработки ЭС

Приведенная последовательность этапов обычно выдерживается в процессе создания «идеального» проекта. В действительности каждый последующий этап разработки способен привнести новые идеи, которые потребуют пересмотра предыдущих решений. Именно поэтому многие специалисты по информатике весьма критично относятся к методологии экспертных систем. Они считают, что расходы на разработку таких систем слишком велики, а полученные в результате программы требуют значительных вычислительных ресурсов.

Выбор проблемы и постановка задачи

Этап выбора проблемы и постановка задачи предваряет процесс разработки ЭС, он включает [Николов и др., 1991]:

- определение проблемной области и задачи;
- поиск эксперта, желающего участвовать в решении проблемы, и формирование коллектива разработчиков;
- определение предварительного подхода к решению проблемы;
- анализ возможных расходов от разработки и определение потенциальной прибыли;
- подготовку подробного плана разработки.

Правильный выбор проблемы представляет критический этап разработки в целом. Если проблема выбрана неправильно, может возникнуть значительное количество задач, для которых эксперт не сможет сформировать однозначной последовательности шагов решения. «Неподходящая» проблема может также привести к разработке экспертной системы, но расходы на создание значительно превысят предполагаемую экономию от ее внедрения. Дело будет обстоять еще хуже, если разработать функционирующую, но неприемлемую для пользователей систему. Даже если разработка выполняется организацией самостоятельно и для внутренних целей, описываемый этап является подходящим моментом для проведения экспертизы на предмет возможности осуществления всего проекта и достижимости поставленных целей.

При выборе области применения ЭС следует учитывать, что если знания, необходимые для решения задач, неизменны, четко формулируются и связаны с вычислительной обработкой, то использование традиционных алгоритмических программ, по всей вероятности, будет наиболее целесообразным способом решения проблем.

Экспертная система ни в коем случае не заменяет реляционные базы данных, статистическое программное обеспечение, электронные таблицы и системы текстовой обработки. Но если результативность решения задачи зависит от трудноалгоритмизуемого знания, которое является интуитивным, изменяющимся, символьным, тогда такие знания можно положить в основу разработки экспертной системы.

Обычно экспертные системы разрабатываются на основании получения специфических знаний от эксперта и ввода их в систему, следовательно, поиск подходящего эксперта — это ключевой шаг в создании экспертных систем.

В процессе разработки и последующего расширения системы инженер по знаниям и эксперт обычно работают совместно. Первоначально они должны понять, будет ли их сотрудничество успешным, поскольку обеим сторонам придется работать совместно, по меньшей мере, в течение одного года. Инженер по знаниям помогает эксперту структурировать знания, определять и формализовать понятия и правила, необходимые для решения проблемы.

Возможные способы программной реализации задачи определяются исходя из характеристик задачи и ресурсов, выделенных на ее решение. Инженер по знаниям предлагает обычно несколько вариантов, связанных с использованием имеющихся на рынке программных средств. Окончательный выбор возможен лишь на этапе разработки прототипа.

На начальном этапе работы инженер по знаниям должен убедиться, что на перечисленные ниже вопросы будут получены утвердительные ответы.

- Данная задача может быть решена с помощью экспертной системы?
- Экспертную систему можно создать имеющимися средствами?
- В коллектив входит подходящий эксперт?
- Предложенные критерии производительности разумны?
- Затраты и срок их окупаемости приемлемы для заказчика?

Затем он составляет план, в котором поэтапно должны быть расписаны процесс разработки и необходимые затраты, а также ожидаемые результаты.

Развитие прототипа до промышленной ЭС

В ходе анализа функционирования прототипа эксперт и инженер по знаниям имеют возможность оценить, какая часть БЗ будет включена в разработку окончательного варианта системы.

Если первоначально выбранные объекты или свойства оказываются неподходящими, их следует заменить. Можно оценить общее число эвристических правил, необходимых для создания окончательного варианта экспертной системы. Иногда при разработке промышленной и/или коммерческой системы выделяют дополнительные этапы для перехода: демонстрационный прототип — действующий прототип — промышленная система — коммерческая система (табл. 1.). Однако чаще происходит плавный переход от демонстрационного прототипа к промышленной системе. В отечественной литературе обычно понятие «коммерческая система» входит в понятие «промышленный программный продукт», или «промышленная ЭС».

После определения основной структуры знаний ЭС инженер по знаниям приступает к разработке и адаптации интерфейсов, с помощью которых система будет взаимодействовать с пользователем и экспертом. Необходимо обратить особое внимание на языковые возможности интерфейсов, их простоту и удобство для управления работой ЭС. Система должна обеспечивать пользователю возможность легким и естественным образом уточнять непонятные моменты, приостанавливать работу и т.д. В частности, могут оказаться полезными графические средства представления знаний.

Таблица 1. Эволюция ЭС от прототипа к промышленной системе

Этап разработки	Функция
Демонстрационный прототип ЭС	Система решает часть задач, демонстрируя жизнеспособность подхода (БЗ содержит несколько десятков правил или понятий)
Исследовательский прототип ЭС	Система решает большинство задач, но неустойчива в работе и не полностью проверена (несколько сотен правил или понятий)
Действующий прототип ЭС	Система надежно решает все задачи на реальных примерах, но при сложной задаче требуется много времени и памяти
Промышленная система	Система обеспечивает высокое качество решений при минимизации требуемого времени и памяти; окончательная версия ЭС разрабатывается с использованием более эффективных средств представления знаний
Коммерческая система	Разработка документации и службы поддержки пользователей

На этом этапе разработки большинство экспертов овладевают средствами ввода и редактирования правил. Таким образом, начинается процесс, во время которого инженер по знаниям передает ведущую роль в процессе создания системы эксперту для уточнения и детальной разработки и обслуживания.

Оценка системы

По окончании этапа разработки промышленной ЭС необходимо протестировать ее эффективность. К тестированию широко привлекаются сторонние эксперты с целью апробирования работоспособности системы на различных примерах. Оцениваются, главным образом, точность работы экспертной системы и ее полезность. Оценку можно проводить исходя из критериев:

- *пользователей* (понятность и «прозрачность» работы системы, удобство интерфейса и др.);

- *приглашенных экспертов* (оценка советов-решений, предлагаемых системой, сравнение их с собственными решениями, оценка подсистемы объяснений и др.);
- *коллектива разработчиков* (эффективность реализации; производительность; время отклика; дизайн; широта охвата предметной области; непротиворечивость БЗ; количество тупиковых ситуаций, когда система не может принять решение; анализ чувствительности программы к незначительным изменениям в представлении знаний, весовых коэффициентах, применяемых в механизмах логического вывода, данных и т.п.).

Стыковка системы

На этом этапе осуществляются стыковка экспертной системы с другими программными средствами в среде, в которой она будет работать, и обучение людей, которых она будет обслуживать. Иногда это требует внесения существенных изменений и вмешательства инженера по знаниям или другого специалиста, который сможет модифицировать систему. Под стыковкой подразумевается также разработка связей между экспертной системой и средой, в которой она функционирует.

Когда экспертная система готова, инженер по знаниям должен убедиться в том, что эксперты, пользователи и персонал знают, как эксплуатировать и обслуживать ее. После передачи своего опыта в области информационной технологии инженер по знаниям может полностью предоставить ЭС в распоряжение пользователей.

Для подтверждения полезности системы важно дать возможность каждому из пользователей поставить перед ЭС реальные задачи, а затем проследить, как она их решает. Для того чтобы система была одобрена, необходимо продемонстрировать ее ассистирующую функцию, освобождающую пользователя от обременительных задач, но не заменяющую. Стыковка предполагает обеспечение связи ЭС с существующими базами данных и другими системами предприятия, а также улучшение системных показателей, зависящих от времени, чтобы можно было обеспечить ее более эффективную работу.

Поддержка системы

При перекодировании системы на язык, подобный Си, повышается ее быстродействие и увеличивается переносимость, однако гибкость при этом снижается. Это приемлемо лишь в том случае, если система сохраняет все знания проблемной области, в предположении, что БЗ не будет изменяться в ближайшем будущем. Однако если экспертная система разрабатывается для динамично изменяющейся проблемной области, то необходимо поддерживать систему в ее инструментальной среде разработки.

Пример 1.3. Классическим примером ЭС, внедренной таким образом, является XCON (R1) в программной среде OPS5 — ЭС, которую фирма DEC использовала для комплектации ЭВМ семейства VAX. Одной из ключевых проблем, с которой столкнулась фирма DEC, была необходимость постоянного внесения изменений для новых версий оборудования, новых спецификаций и т.д.

Технология быстрого прототипирования

Важнейшую роль при разработке ЭС играет технология прототипирования. Прототип системы является усеченной версией экспертной системы, спроектированной для проверки правильности кодирования фактов, связей и стратегий рассуждения эксперта. Он также позволяет инженеру по знаниям привлечь эксперта к активному участию в процессе разработки экспертной системы и, следовательно, к принятию им обязательства приложить все усилия к созданию системы в полном объеме.

Прототип содержит несколько десятков записей на языке представления знаний (ЯПЗ): правил, фреймов или примеров. На рис. 6 приведены шесть стадий разработки прототипа и указан минимальный коллектив разработчиков, занятых на каждой из стадий [Хейес-Рот и др., 1987; Джексон, 2001]. Охарактеризуем кратко каждую стадию.

Идентификация проблемы

На этом этапе члены коллектива разработчиков знакомятся друг с другом, происходит их обучение, вырабатывается неформальная формулировка проблемы, уточняется задача, планируется ход разработки прототипа экспертной системы, также определяются:

- необходимые ресурсы (время, люди, компьютеры и т.д.);
- источники знаний (книги, дополнительные эксперты, методики);
- имеющиеся аналогичные экспертные системы;
- цели (распространение опыта, автоматизация рутинных действий и др.);
- классы решаемых задач и т.д.

Извлечение знаний

На этом этапе инженер по знаниям получает наиболее полное из возможных представлений о предметной области и способах принятия решения. Происходит обучение инженеров по знаниям с использованием различных методов: анализ текстов; диалоги; экспертные игры; лекции; дискуссии; интервью; наблюдение и др.

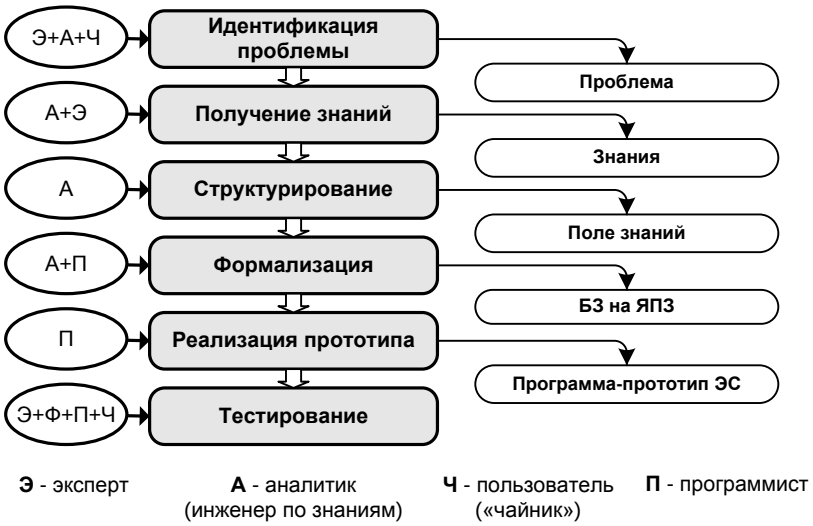

Рисунок **6 - Стадии разработки прототипа ЭС**

Структурирование, или концептуализация, знаний

Разрабатывается неформальное описание знаний о предметной области в виде графа, таблицы, диаграммы или текста, которое отражает основные концепции и взаимосвязи между понятиями предметной области. Такое описание называется полем знаний. Выявляется структура полученных знаний о предметной области, т.е. определяются:

- терминология;
- список основных понятий и их атрибутов;
- отношения между понятиями;
- структура входной и выходной информации;
- стратегия принятия решений;
- ограничения стратегий и т.д.

Формализация знаний

Создается формализованное представление концепций предметной области на основе выбранного ЯПЗ. Выполняется разработка БЗ на языке представления знаний, который, с одной стороны, соответствует структуре поля знаний, а с другой — позволяет реализовать прототип системы на следующей стадии программной реализации. Традиционно на этом этапе используются:

- логические методы (исчисления предикатов 1-го порядка и др.);
- продукционные модели (с прямым и обратным выводом);
- семантические сети;
- фреймы;
- объектно-ориентированные языки, основанные на иерархии классов и объектах;
- онтологии.

Реализация

Это окончательный этап разработки программного комплекса, демонстрирующий жизнеспособность подхода в целом. Создается прототип экспертной системы, включающий базу знаний и остальные блоки, при помощи одного из следующих способов:

- программирование на традиционных языках типа PYTHON, C#, Java и др.;
- программирование на специализированных языках, применяемых в задачах искусственного интеллекта — LISP, ПРОЛОГ и др.;
- использование инструментальных средств разработки ЭС типа CLIPS, DROOLS и др.;
- использование «пустых» ЭС («оболочек») или программных комплексов типа EXSYS, G2 и др.

Перед разработчиками ЭС всегда остро стоит вопрос о выборе адекватного программного инструментария разработки. Быстро изменяющийся рынок средств разработки предлагает десятки программных систем, однако при выборе надо принимать во внимание целый ряд как объективных, так и субъективных факторов. Основные факторы, влияющие на выбор программного инструмента:

- особенности предметной области;
- наличие квалифицированной команды разработчиков;
- необходимость стыковки с другими программными продуктами;
- бюджет проекта;
- количество пользователей будущей системы и их квалификация.

Особенной популярностью пользуются «пустые» ЭС, или «оболочки», из-за простоты и скорости разработки БЗ. Свободно распространяемые и недорогие «оболочки»:

- E2glite http://www.expertise2go.com/webesie/e2gdoc/;
- FOCL ics.uci.edu:/pub/machine-learning-programs/;
- BABYLON ftp.gmd.de:/gmd/ai-research/Software/Babylon/;
- MIKE www.hcrl.open.ac.uk;
- MIKEv2.50: pub/software/pc/MIKEV25.ZIP;
- JESS http://herzberg.ca.sandia.gov/jess/download.shtml;

Коммерческие «оболочки»:

- ACQUIRE;
- Angoss Knowledge Seeker;
- First Class;
- Knowledge Craft;
- Arity Expert Development Package;
- Runner.

Отдельно следует упомянуть ЭС реального времени, для разработки которых используются более сложные инструменты. Например, GenSAA — программный продукт, предназначенный для быстрой разработки

экспертных систем реального времени, которые отслеживают и обнаруживают неполадки в сложных программно-аппаратных комплексах.

Тестирование

Этот этап включает выявление ошибок в подходе и реализации прототипа и выработку рекомендаций по доводке системы до промышленного варианта. Оценивается и проверяется работа программ прототипа с целью приведения в соответствие с реальными запросами пользователей. Проверяются следующие свойства:

- удобство и эргономичность интерфейсов ввода–вывода (характер вопросов в диалоге, связность выводимого текста результата и др.);
- эффективность стратегии управления (порядок перебора, использование нечеткого вывода и др.);
- качество проверочных примеров;
- корректность базы знаний (полнота и непротиворечивость правил).

Продукционная модель и архитектура продукционной системы

Основными компонентами архитектуры продукционной системы являются (рис. 7):

- БЗ продукционных правил;
- рабочая память;
- цикл управления распознавание–действие.

Моделирование решения задачи основано на процессе *сопоставления с образцом* (pattern matching), в ходе которого текущее состояние решения сравнивается с имеющимися знаниями для определения дальнейших действий.

В БЗ содержится множество *продукционных правил* или просто *продукций* (productions), являющихся парами условие–действие, которые определяют элементарные шаги решения задачи. *Условная часть* (IF–part) правила является шаблоном (образцом), по которому можно определить, в какой момент необходимо использовать (активировать) данное правило для выполнения очередного этапа решения задачи. *Часть действия* (THEN–part) описывает соответствующий шаг решения. Условную часть правила также называют *антецедентом* (antecedent), а часть действия — *консеквентом* (consequent).

Рабочая память (working memory) содержит текущее описание модели мира в процессе рассуждений. В этой модели содержится набор образцов, инициализируемый начальным описанием задачи.

В управляющем цикле *распознавание–действие* (the recognize — act cycle) осуществляется сравнение образцов из рабочей памяти с условными

частями правил в БЗ. Если условие какого-либо правила соответствует образцу, то это правило помещается в *конфликтное множество* (conflict set). Продукции, содержащиеся в конфликтном множестве, называют *допустимыми*, так как они согласованы с текущим состоянием рабочей памяти. После того, как закончит работу цикл распознавание–действие, осуществляется процесс *разрешения конфликтов* (conflict resolution), в ходе которого выбирается и активизируется (возбуждается) одна из допустимых продукций. Наконец, в соответствии с частью действия активированного правила осуществляется модификация рабочей памяти. Весь этот процесс повторяется до тех пор, пока образцы в рабочей памяти не будут соответствовать ни одному из правил БЗ.

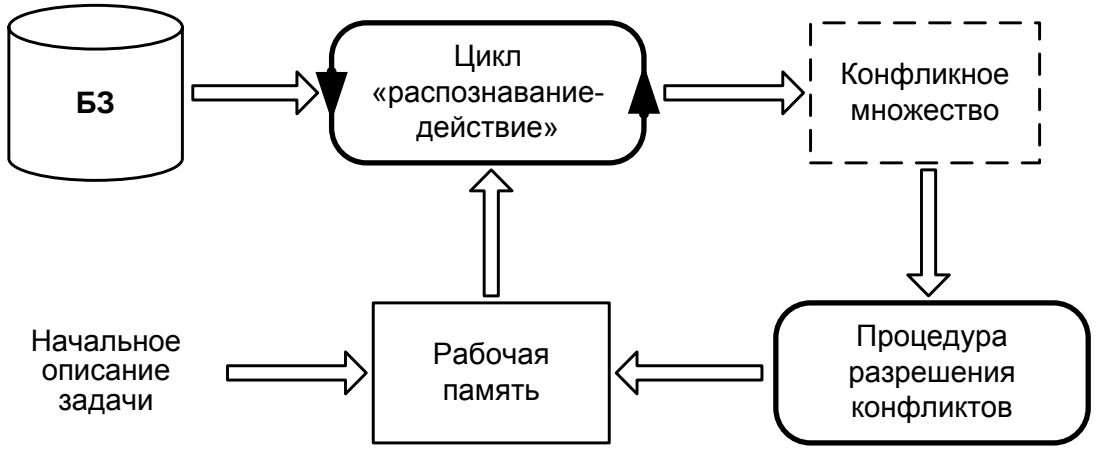

Рисунок 7- Архитектура продукционной системы

Стратегии разрешения конфликтов отличаются в различных реализациях продукционной модели и могут быть достаточно простыми. Например, выбирается первое из допустимых правил. Однако многие системы допускают использование сложных эвристик для осуществления выбора из конфликтного множества. Например, в системе OPS5 поддерживаются следующие стратегии разрешения конфликтов:

1. *Рефракция* (refraction) для предотвращения зацикливания: после активизации правила оно не может быть использовано снова, пока не измениться содержимое рабочей памяти.
2. *Новизна* (recency) позволяет сосредоточить поиск на одной линии рассуждения: предпочтение отдается правилам, в условии которых встречаются факты, добавленные в рабочую память последними.
3. *Специфичность* (specifity) отдает предпочтение более конкретным правилам перед более общими: одно правило более специфично (конкретно), чем другое, если оно содержит больше фактов в условной части.

Чистая продукционная модель не предусматривает выхода из тупиковых ситуаций в процессе поиска. Работа продолжается, пока не

исчерпаны все возможные варианты. Гораздо более эффективным является модификация цикла управления с механизмом *возврата* (cycle back) в предыдущее состояние модели мира, позволяющего находить более точные решения задачи.

Продукционная модель позволяет имитировать различные аспекты поведения человека при решении задач. БЗ продукций соответствует навыкам решения задач в *долговременной памяти*. Подобно знаниям в долговременной памяти продукционные правила не изменяются в процессе решения задачи. Новые навыки могут быть просто добавлены в БЗ при необходимости. Рабочая память соответствует *кратковременной памяти*. В ходе решения задачи «фокус внимания» переходит от одного шага к другому, при этом после получения решения содержимое рабочей памяти не сохраняется.

В качестве *условия* и *действия* в правилах может быть, например, предположение о наличии того или иного свойства, принимающее значение *истина* или *ложь*. При этом термин *действие* следует трактовать широко: это может быть директива к выполнению какой-либо операции, рекомендация, или модификация базы знаний — предположение о наличии какого-либо производного свойства. Примером продукции может служить следующее выражение:

П1: *ЕСЛИ* клиент работает на одном месте более двух лет,
ТО клиент имеет постоянную работу.

Как условие, так и действие правила могут учитывать несколько выражений, объединенных логическими связками *И, ИЛИ, НЕ*:

П2: *ЕСЛИ* клиент имеет постоянную работу
И клиенту более 18 лет
И клиент НЕ имеет финансовых обязательств,
ТО клиент может претендовать на получение кредита.

Помимо продукционных правил база знаний должна включать и простые *факты*, поступающие в систему через интерфейс пользователя или выводимые в процессе поиска решения задачи. Факты являются простыми утверждениями типа «клиент работает на одном месте более двух лет». Факты, являясь истинными утверждениями, копируются в рабочую память для использования в цикле распознавание–действие.

Последовательное активирование правил порождает *цепочку вывода* (reasoning). Цепочка вывода, полученная в результате последовательного выполнения правил П1 и П2 показана ниже (рис 8.). Эта цепочка показывает, как на основании правил и исходных фактов выводит

заключение о возможности получения кредита. В ходе первой итерации вывода управляющий цикл, сопоставляя факт «клиент работает на одном месте более двух лет» и правило П1 добавляет в рабочую память новый факт (образец) «клиент имеет постоянную работу». Далее на основании содержимого рабочей памяти и правила П2 выводится новый факт «клиент может претендовать на получение кредита», который и является окончательным решением, так как цикл распознавание–действие останавливается и процесс вывода на этом завершается.

Монотонным выводом в продукционных системах называют вывод, при котором факты не удаляются из рабочей памяти. *Немонотонный вывод* допускает удаление фактов из рабочей памяти. При немонотонном выводе существенную роль играет порядок применения продукционных правил.

Цепочки вывода экспертной системы могут быть предъявлены пользователю и помогают понять, как было получено решение. Также визуализация этого процесса может быть полезна в процессе отладки БЗ, так как эксперт получает возможность «увидеть» неправильный ход рассуждений. Большинство реализаций продукционных систем имеют встроенные средства для интроспекции БЗ и мониторинга работы машины вывода.

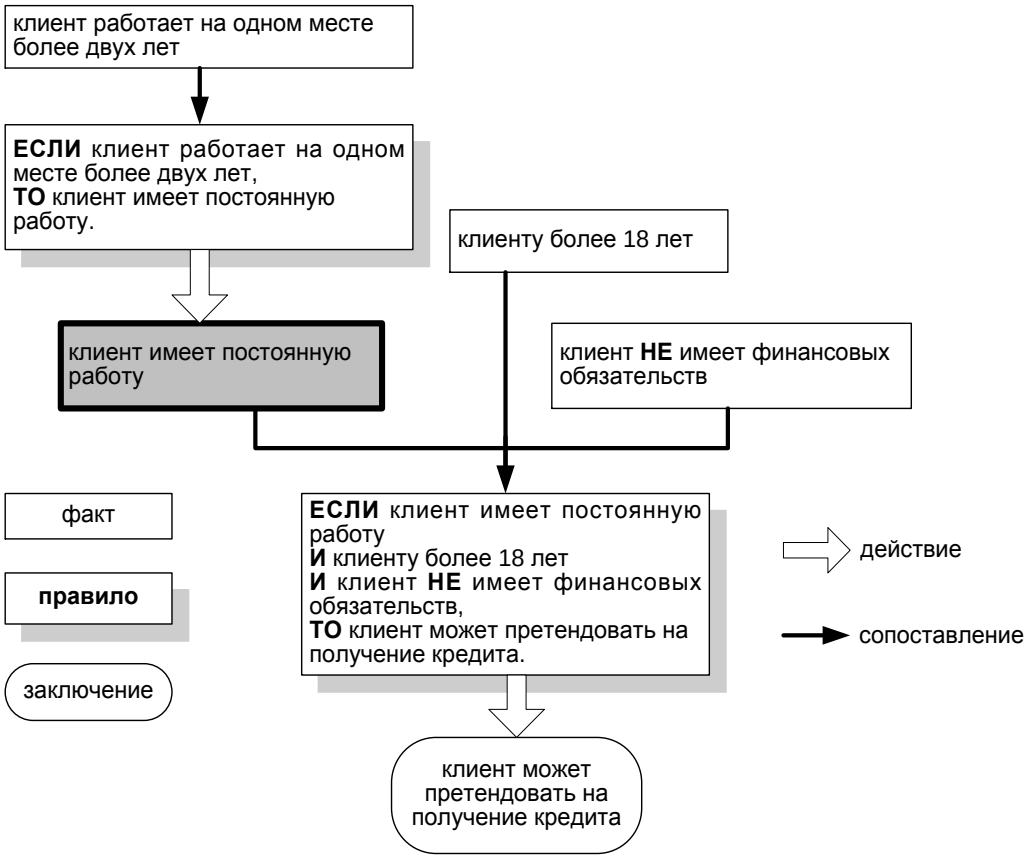

Рисунок 8 - Пример цепочки вывода в продукционной системе

Вывод решения может осуществляться в двух направлениях: от данных задачи к цели и в обратном направлении от цели к исходным данным. *Вывод на основе данных* (data–driven search), процесс решения задачи начинается с исходных фактов. Затем, применяя допустимые правила, осуществляется переход к новым фактам. И так до тех пор, пока цель не будет достигнута. Это процесс также называют *прямой цепочкой вывода* (forward chaining). *Вывод от цели* (goal–directed strategy) начинается от одной из допустимых целей, и рассматриваются пути, ведущие к достижению этой цели. Таким образом, определяется последовательность правил, позволяющих найти решение. Процесс повторяется для всех заданных в задаче целей. Такой способ поиска называют также *обратной цепочкой вывода* (backward chaining). Выбор направления поиска зависит от конкретной задачи, структуры исходных данных и целей, способов реализации алгоритмов поиска. Следует отметить, что, как правило, в реальных системах используются комбинации обоих способов вывода.

Лабораторная работа № 1. Создание модели базы знаний

Первой задачей в разработке базы знаний, является создание модели, описывающей факты (объекты, в терминах объектно-ориентированного подхода), которые участвуют в описании бизнес-правил.

В нашем примере определим два факта: заявление на кредит, решение о выдаче кредита. А так как для наименования фактов необходимо использовать только латинский алфавит, то для каждого факта придумаем название: заявление на кредит – ApplicationForCredit и решение о выдачи кредита – CreditDecision.

ApplicationForCredit будет содержать следующие поля:

- Сумма кредита (AmountOfCredit) – сумма в рублях, которую запрашивает заемщик;
- Срок кредитования (PeriodOfCredit) – срок в месяцах, на который заемщик запрашивает кредит;
- Ежемесячный доход (Salary) – заработная плата заемщика в рублях;
- Возраст (Age) – возраст заемщика в годах;
- Пол (Sex) – пол заемщика, М или Ж;
- Опыт работы (JobExperience) – совокупный опыт работы заемщика в годах;
- Последний срок работы (LastPeriodOfWork) – срок работы заемщика на последнем рабочем месте в месяцах;
- Сумма текущих обязательств (CurrentObligations) – сумма в рублях, которую заемщик выплачивает ежемесячно по другим кредитам;

CreditDecision будет содержать следующие поля:

- Ответ (Decision) – решение банка: «отказать в кредите» или «выдать кредит» (значение по умолчанию);
- Ежемесячная плата (MonthlyFee) – ежемесячная плата по кредиту в рублях;

Создание модели

Для того чтобы создать модель выделим пакет, который будет содержать все бизнес-правила, модель и тестовые сценария для базы знаний, назовём его пакет «CreditKB» нажимаем «Create New» и выбираем «New Declarative Model». Далее в открывшемся окне вводим имя модели, выбираем пакет и нажимаем «OK», см. рисунок 9.

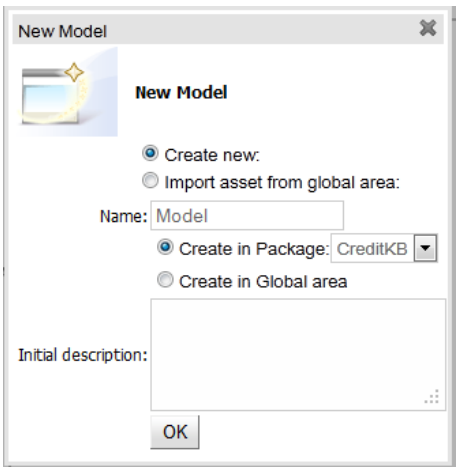

Рисунок 9 - Окно создания модели базы знаний

Создание факта

После создания модели откроется вкладка, в которой нажимаем «Add new fact type», вводим имя факта и жмем «OK». Повторим это действие для ApplicationForCredit и CreditDecision, тогда в модели должно появиться два факта как на рисунке 10.

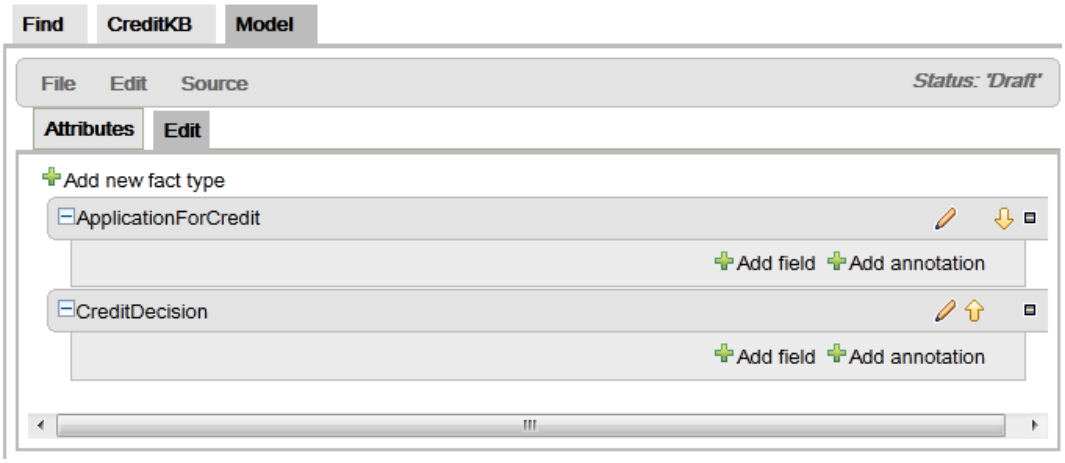

Рисунок 10 - Список фактов в модели Model

Создание поля

Когда факт создан, мы можем приступить к созданию полей этого факта, для этого нажимаем «Add field» и в открывшемся окне вводим имя поля («Field name») и его тип («Type»).

Типы полей:

- Whole number (integer) – целое число,
- True or False (boolean) – булевое значение, истина или ложно,
- Text (string) – текст,
- Date (java.util.Date) – время и дата,
- Decimal number (java.util.BigDecimal) – число с плавающей точкой.

После того как вы созданите поля два обоих фактов у вас должен получится такой же список как на рисунке 11.

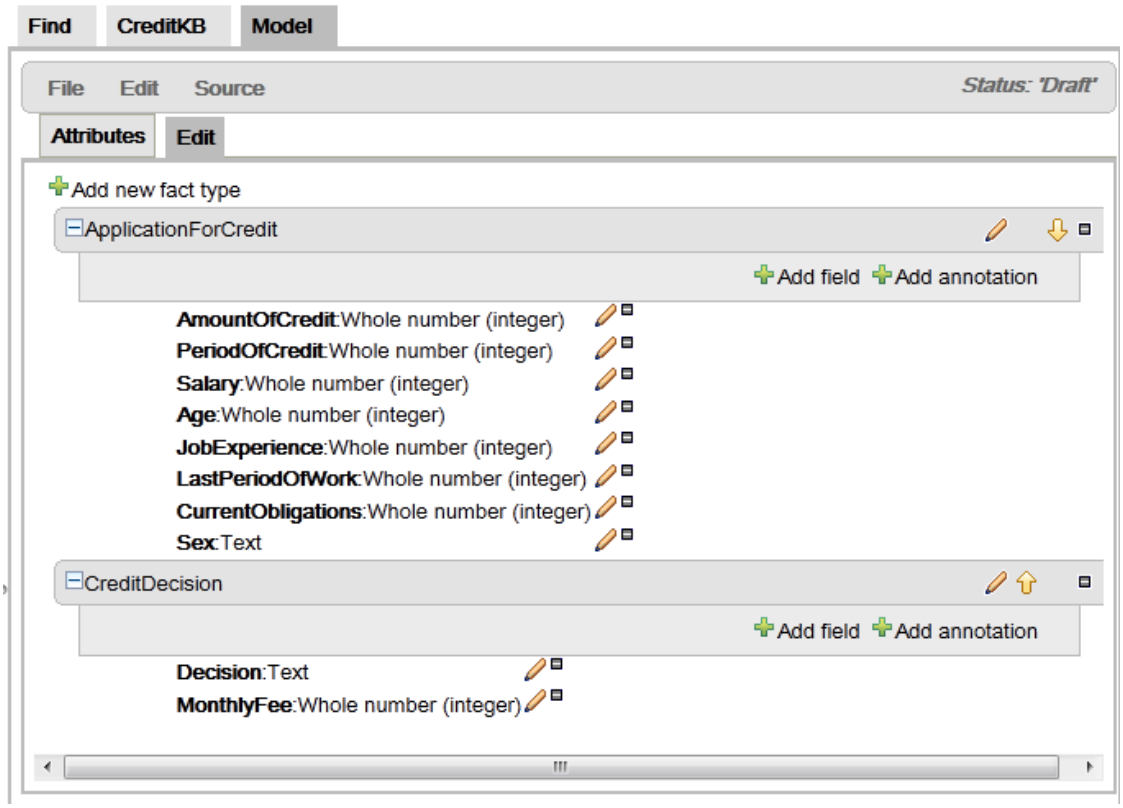

Рисунок 11 - Вкладка модели базы знаний

После того как вы закончили, необходимо сохранить модель, для этого нажмите «File», выбирите «Save changes», в появившемся окне введите комментарий к сделанным изменениям и нажмите «OK».

Если сохранения прошло успешно, то во вкладке пакета «CreditKB» вы увидите созданную модель как на рисунке 12.

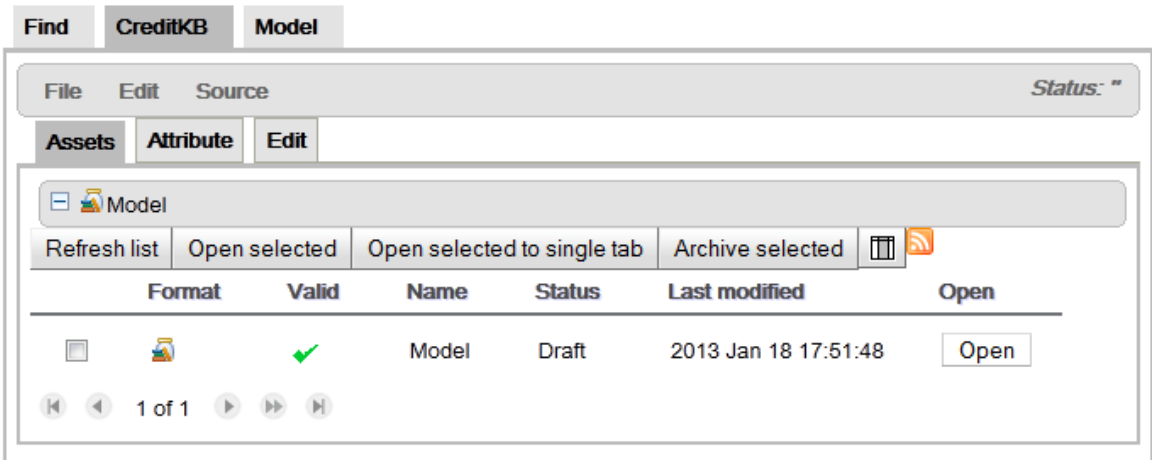

Рисунок 12 - Вкладка с содержимым пакета CreditKB

Теперь вы готовы перейти к описанию правил.

Лабораторная работа № 2. Создание простого правила

База знаний будет состоять всего из пяти правил (это очень мало для реальной базы знаний, но для нашего примера этого количества достаточно).

Создание правила

Первое правило, которое мы опишем, будет следующим:

Если возраст заемщика < 18 лет
То отказать в кредите.

Для этого создадим пустое правило: выделяем пакет, нажимаем «Create New» и выбираем «New Rule», см. рис. 13.

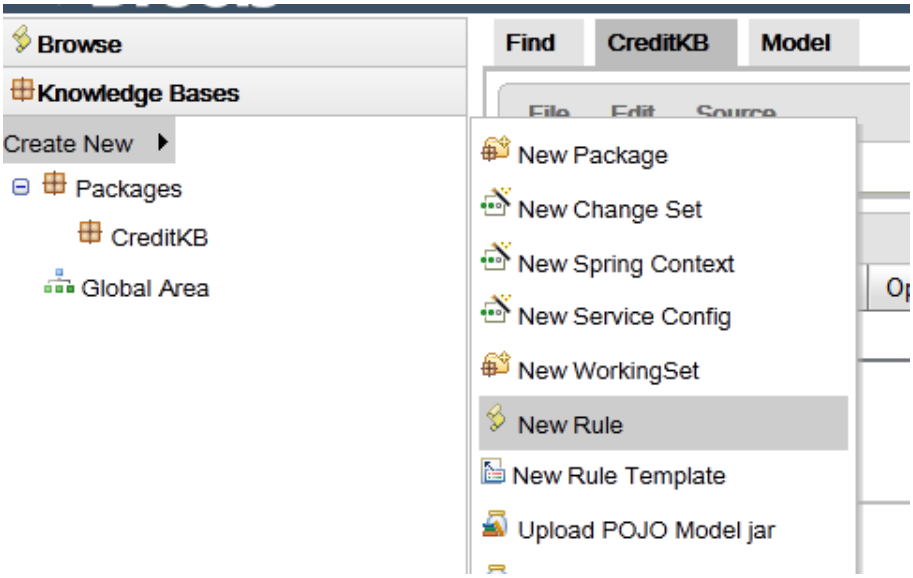

Рисунок 13 - Создание нового правила

В появившемся окне необходимо ввести имя правила и, если нужно, то и описание. И не забудьте, что должен быть выбран пакет, в котором будет создано правило, см. поле «Create in Package» (см. рис. 14).

Для простоты, именем правила было выбрано «Rule #1», но это не является каким либо требованием к наименованию, в качестве имени может выступать любая строка на любом языке.

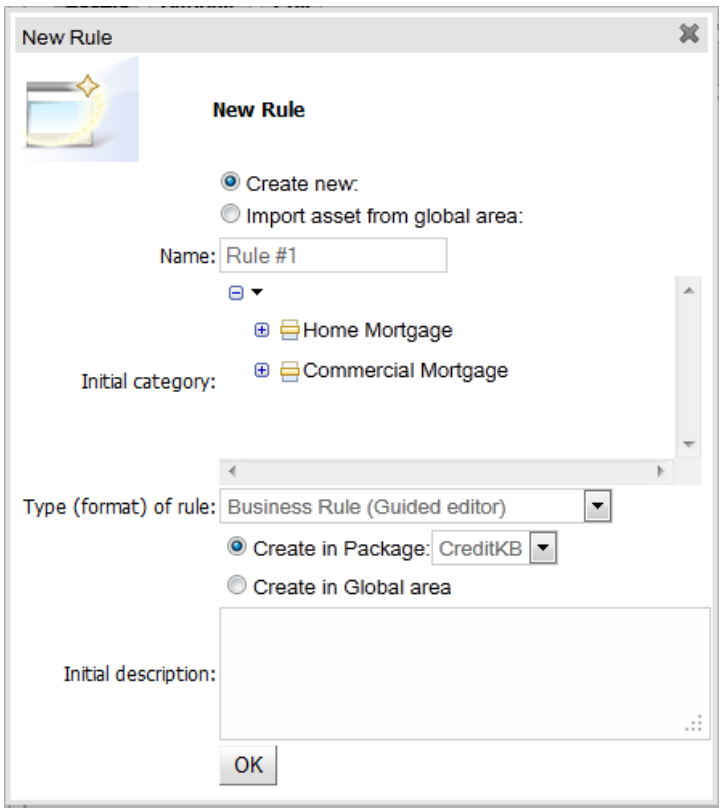

Рисунок 14 - Окно создания правила

После того как правило создано, оно отобразится во вкладке пакета, см. рис. 15.

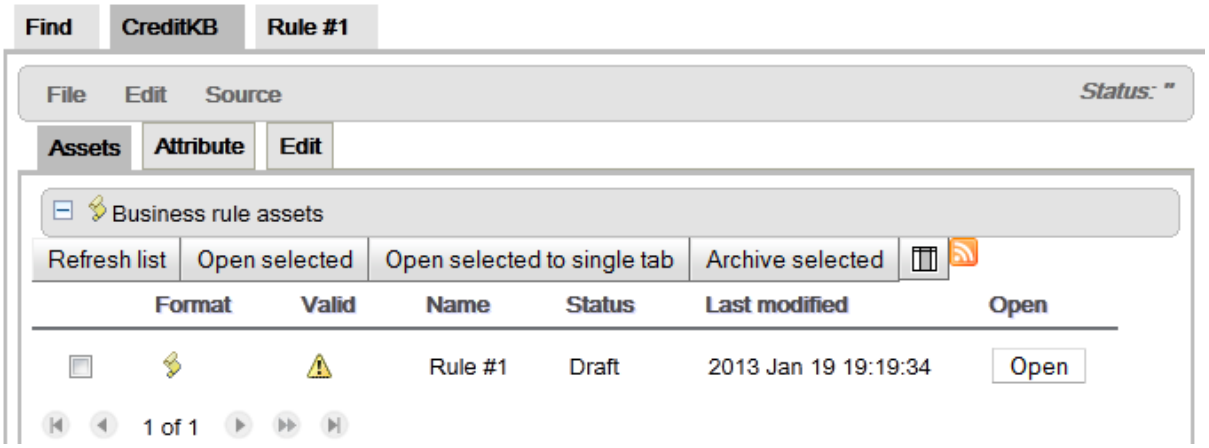

Рисунок 15 - Список правил пакета "CreditKB"

Теперь правило готово для его описания, т.е. для заполнения частей «условия» и «действия», в терминах Drools Guvnor – это When и Then части правила соответственно.

Описание части условий

Для того чтобы описать условие «возраст заемщика меньше 18 лет» необходимо выполнить следующие действия: нажать + напротив слова «WHEN» и в открывшимся окне выбрать ApplicationForCredit, после чего у вас должно появится условие как на рис. 16.

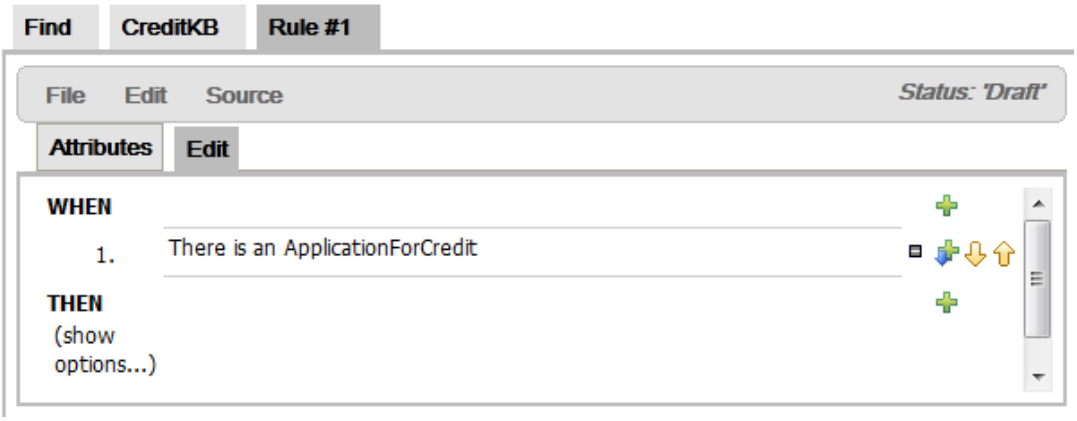

Рисунок 16 – Правило с новым условием

Добавленное условие означает, что правило выполнится, когда в рабочей памяти появится факт типа ApplicationForCredit.

Но это еще не все, теперь нам надо добавить условие «возраст меньше 18 лет» для этого необходимо добавить ограничение на поле факта ApplicationForCredit: нажимаем на название только что добавленного условия и в появившемся окне выбираем в поле «Add a restriction on a field» поле факта Age, см. рис. 17.

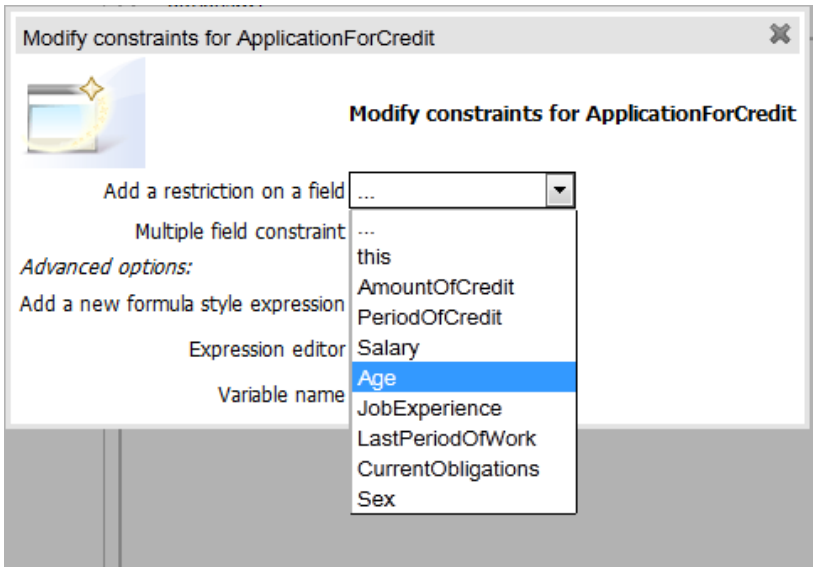

Рисунок 17 - Установка условия на поле факта ApplicationForCredit

Далее мы видим, что было добавлено дополнительное ограничение, рядом с которым есть выпадающий список с дополнительными условиями. В этом списке выбираем условие «less than», нажимаем рядом нарисованный карандаш, и в появившемся окне нажимаем «Literal Value», после чего вместо карандаша появляется пусто поле, в которое записываем число 18, см. рис. 18.

Теперь описание условия «возраст заемщика меньше 18 лет» закончено.

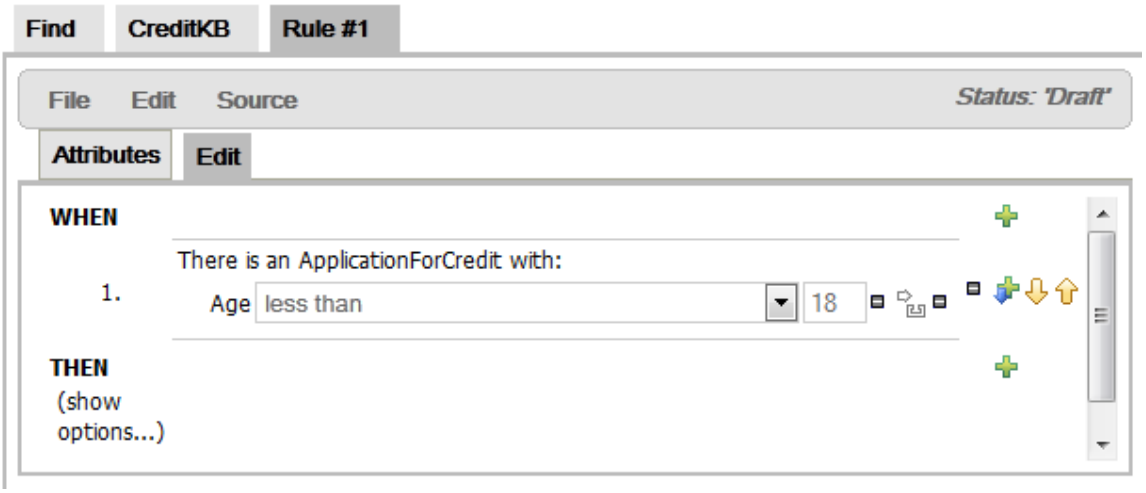

Рисунок 18 - Добавленное ограничение на поле Age

При срабатывании этого правила необходимо указать, что выдача кредита отказана, т.е. записать в поле Decision факта CreditDecision слова «отказать в кредите», а что бы это реализовать, сначала необходимо добавить в часть WHEN условие наличия факта CreditDecision, в котором будет записан текст «отказать в кредите» при срабатывании правила, и присвоить его переменной, назовем её «decision».

Действия: нажимаем значок плюса со стрелкой и в появившемся окне выбираем CreditDecision и нажимаем «OK». Теперь для присвоения переменной нажимаем на только что добавленное условие, в открывщемся окне вводим в поле «Variable name» имя переменной и нажимаем «Set». После чего правило должно выглядить так как показано на рис. 19.

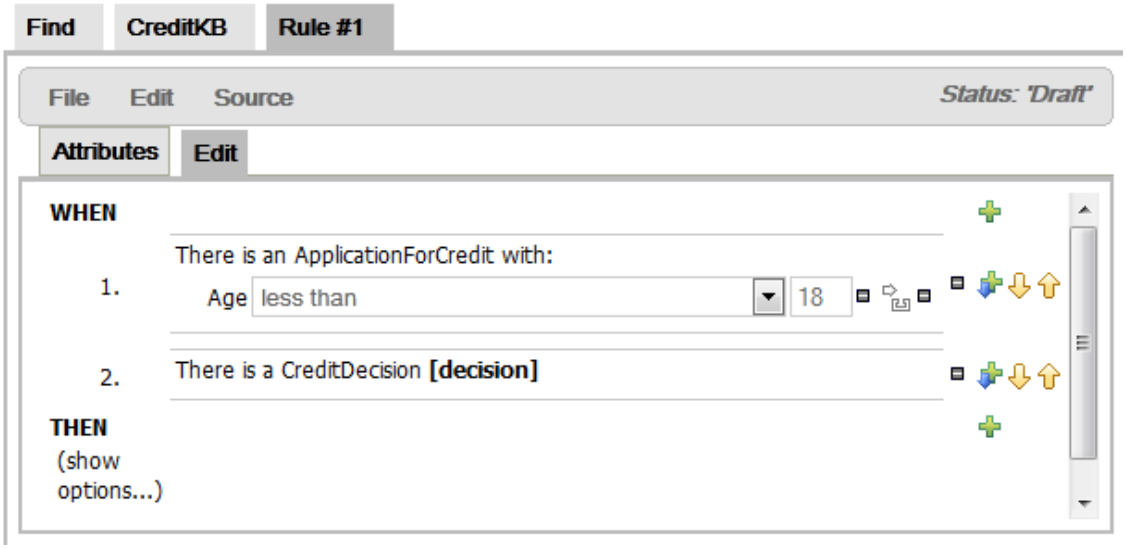

Рисунок 19 - Присвоение факта переменной

На этом мы закончили с описанием части WHEN и можем перейти к описанию следующей части правила.

Описание части действий

Как было написано выше, необходимо записать в поле Decision факта CreditDecision, который присвоен переменной decision, текст «отказать в кредите».

Действия: нажимаем значок плюса в части THEN и в появившемся окне выбираем «Change field values of decision», где «decision» – это имя созданной нами переменной, и нажимаем «OK», в результате чего появляется рисунок карандаша. Далее по нажатию на появившийся рисунок карандаша, открывается окно, в котором необходимо выбрать в списке значение «Decision», которое является именем одноименного поля факта. И снова нажимаем на рисунок карандаша, выбираем «Literal Value» и вводим текст «отказать в кредите».

В результате выполненых действий правило должно выглядить как на рис. 20.

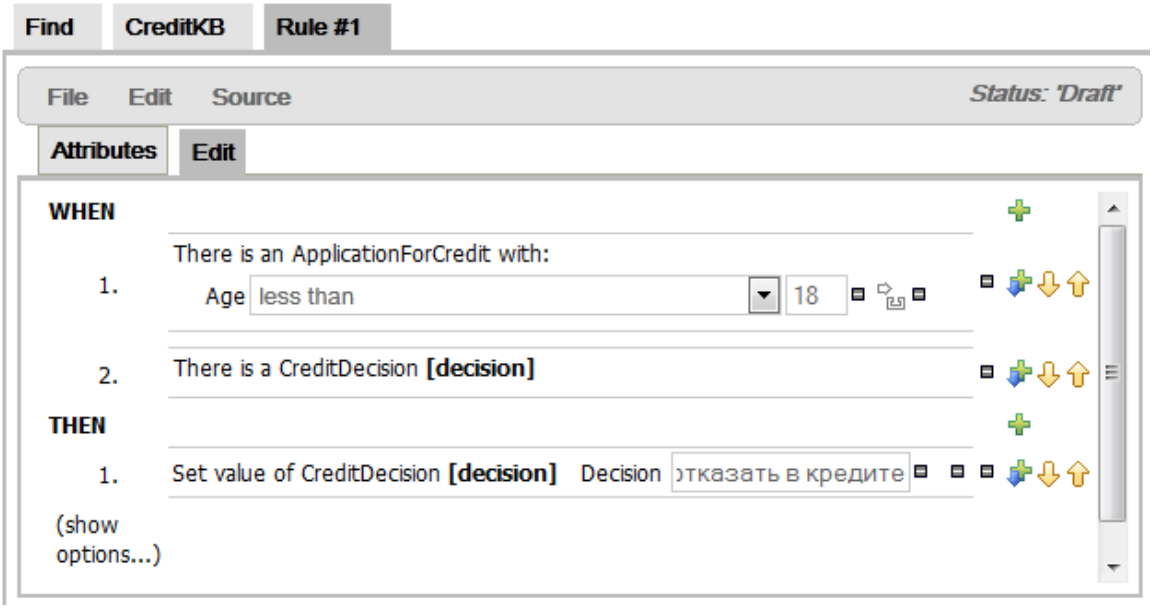

Рисунок 20 - Законченное правило

Описание правила закончено, но перед тем как мы сохраним его необходимо проверить, что оно успешно проходит валидацию и верификацию. Для этого поочередно запускаем команды «Validate» и «Verify». Если валидация и верификацию прошли успешно, то мы готовы сохранить правило, сохранение происходит командами «Save changes» или «Save and close».

Отлично! Теперь вы умеете создавать простые правила и готовы перейти к более сложным их вариантам.

Лабораторная работа № 3. Создание правила: формулы

В этой лабораторной работе мы создадим более сложное правило, которое использует конструкцию формул для вычисления различнх выражений.

Разделение на простые и сложные правила достаточно условное, так как в целом структура и работа с ними одинакова, единственное отличие – это использование конструкций более гибких и в тоже время требующих более глубокого понимания.

Использование формул в части условий

Правило:

Если возраст + срок кредитования >= 60 лет для мужчин и 55 лет для женщин
То отказать в кредите.

Разобъем условие правила на два подусловия, которые объединим логической операцией ИЛИ:

- заемщик – мужчина и его возраст + срок кредитования >= 60 лет

- заемщик – женщина и её возраст + срок кредитования >= 55 лет

Для того чтобы объединить несколько условий операцией ИЛИ используется конструкция «Any of the following are true…», см. рис. 21.

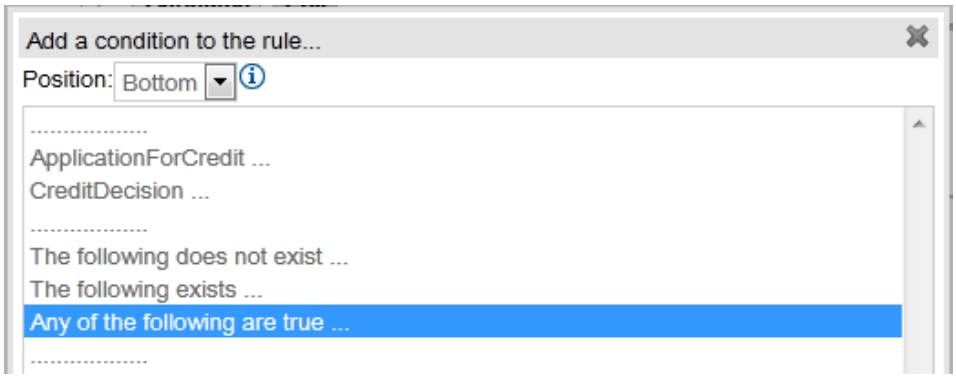

Рисунок 21 - Выбор конструкции "Any of the following are true..."

Использование операции ИЛИ означает что объединенное условие истино тогда когда хотя бы одно из подусловий истино.

Создадим новое правило с именем Rule #2 и запишем условие, но пока только с указанием пола заемщика, а сравнение суммы возраста и срока кредитование для простоты пока опустим, см. рис. 22.

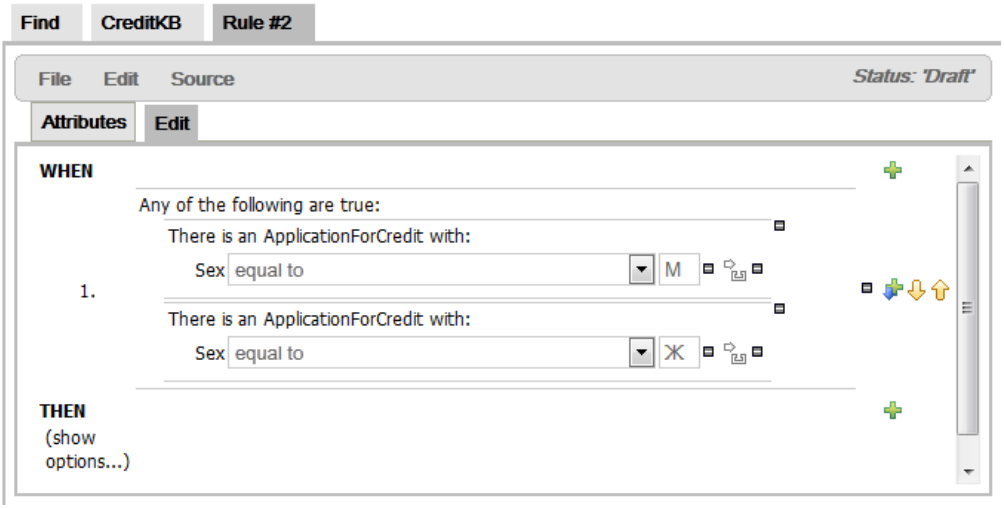

Рисунок 22 - Частично записанное условие правила Rule #2

Замечательно! Следующий шаг – это использование формул для вычисления различных выражений, которые используются при описании условий.

Что закончить описание условия необходимо добавить формулу, которая вычисляет сумму полей Age и PeriodOfCredit и сравнивает её с числом 60 для первого подусловия и с числом 55 для второго подусловия.

Действия: нажимаем на «There is an ApplicationForCredit with:» из первого подусловия и в открывщемся окне нажимаем «New formula», в результате чего появляется поле для записи формул, в которое записываем следующее выражение: Age + PeriodOfCredit / 12 >= 60.

Повторив теже действия и для второго подусловия, мы получим то же самое, что и на рис. 23.

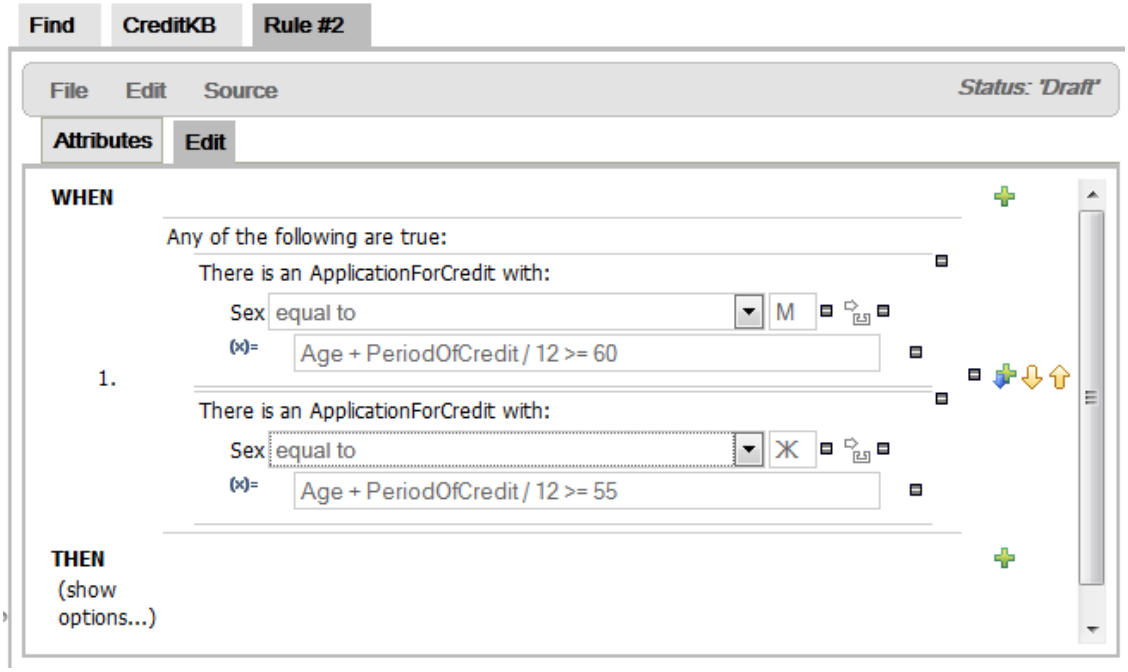

Рисунок 23 - Правило использующее конструкцию "New formula"

Теперь в правило осталось добавить действие, которое, как вы уже догадались, идентично действию из правила предыдущей лабораторной работы.

Законченное правило Rule #2 показано на рис. 24.

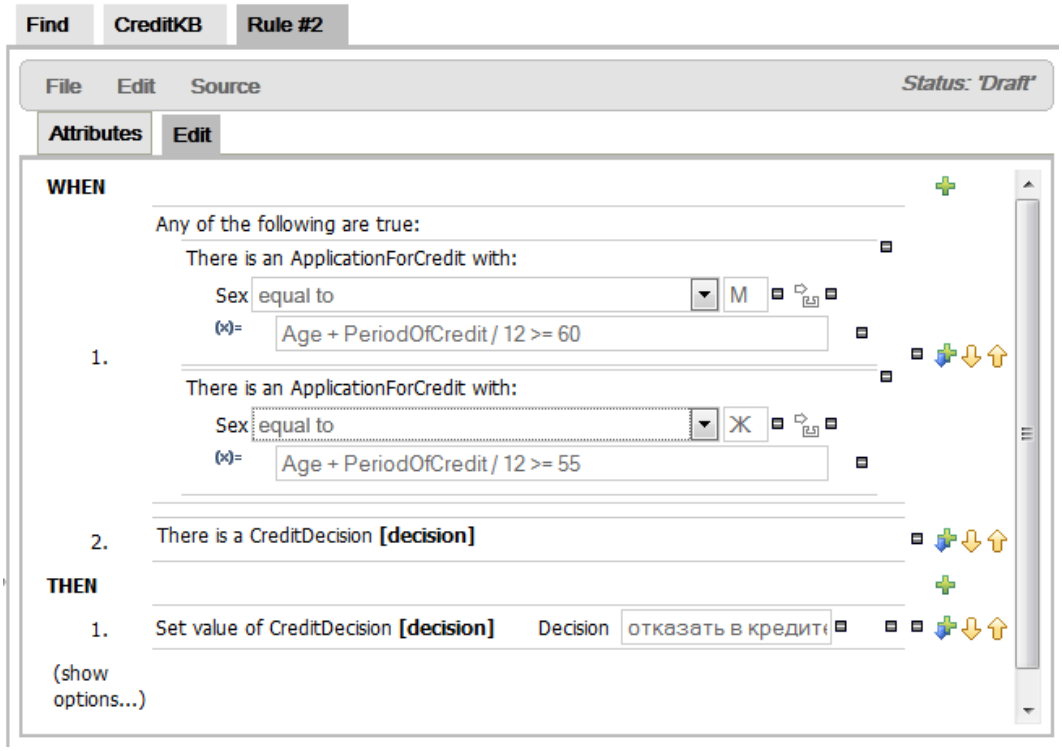

Рисунок 24 - Законченное правило Rule #2

Использование формул в части действий

В следующем правиле мы используем конструкцию формул для вычисления ежемесячного платежа по кредиту:

Если есть заявка на кредит
То рассчитать ежемесячный платеж по кредиту.

По своей сути, это правило не несет никакой смысловой нагрузки, а лишь является способом вычисления и записи ежемесячного платежа в поле MonthlyFee факта CreditDecision.

Пусть годовая процентная ставка будет равна 10%, а платежи по кредиту будут аннуитетными, т.е. платеж всегда одинаков, то ежемесячный платеж вычисляется по следующей формуле:

$$ЕП = СК * (ПС + \frac{ПС}{(1+ПС)^{П}-1}),$$

где ЕП – ежемесячный платеж, СК – сумма кредита, ПС – месячная процентная ставка, которая равна 1/12 от годовой процентной ставки (в нашем случае равно 0.1/12) и П – срок кредитования в месяцах.

Правило Rule #3 см. на рис. 25.

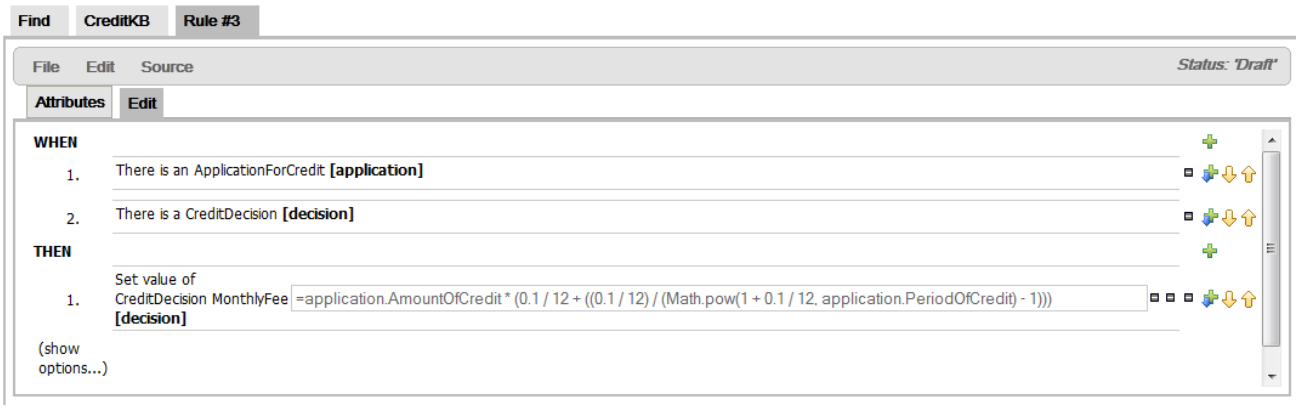

Рисунок 25- Правило Rule #3

Как вы заметили, данное правило будет выполнено тогда и только тогда, когда в рабочей памяти будут находиться факты ApplicationForCredit и CreditDecision. В результате чего в поле MonthlyFee факта CreditDecision, который присвоен переменной «decision», будет записана ежемесячная плата по кредиту.

Вы уже знакомы с тем, как записывать текстовое или числовое значение в поле факта, а если нет, то см. предыдущую лабораторную работу. Для того чтобы вместо текстового или числового значения записать в поле результат вычисления какого-либо выражения необходимо по нажатию на карандаш вместо «Literal value» выбрать «Formula», см. рис. 26, и записать нужное выражение.

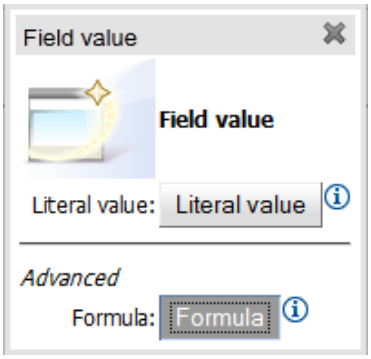

Рисунок 26 - Выбор формулы в качестве значения поля

Хорошо! В этой лабораторной работе вы познакомились с несколькими новыми конструкциями: конструкция ИЛИ («Any of the following are true…») и конструкция формул. И реализовали два более сложных, чем в предыдущей работе, правил.

Лабораторная работа № 4. Создание правила: атрибуты

Осталось ещё два правила, которые нам необходимо создать для полноценной картины, первое из них:

Если ежемесячный доход – сумма текущих обязательств – ежемесячный платеж по кредиту < 10% от ежемесячного дохода
То отказать в кредите.

Теперь вы знаете, как создавать правила и использовать формулы, поэтому я приведу только готовый вариант правила, см. рис. 27.

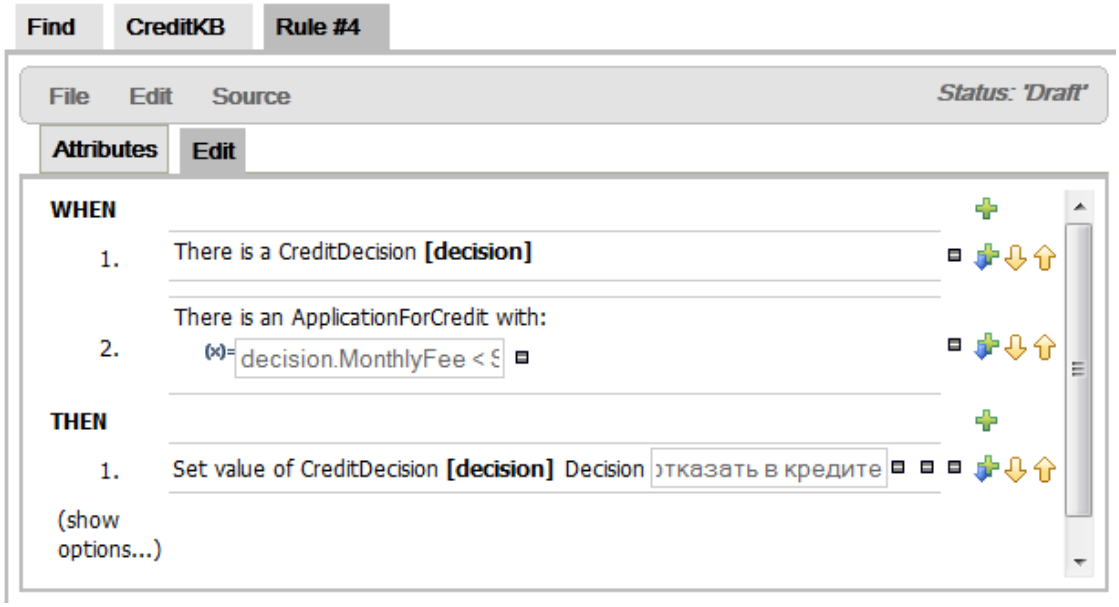

Рисунок 27 - Правило Rule #4

В части условий правила Rule #4 записана следующая формула:

Salary - CurrentObligations - decision.MonthlyFee < Salary * 0.1

Как видно из формулы, для вычисления условия используется поле MonthlyFee факта CreditDecision, в которое должна быть записана ежемесячная плата по кредиту, которая в свою очередь вычисляется в правиле Rule #3. Что если правило Rule #4 выполнится раньше, чем Rule #3?

Результат будет непредсказуем, поэтому необходимо гарантировать, что правило Rule #3 выполнится раньше, т.е. будет иметь больший приоритет, чем правило Rule #4. Что бы этого добится будем использовать такой атрибут правила как «salience», который устанавливает приоритет правила, по умолчанию у всех он равен 0, для правила Rule #3 установим его на 10 (или любое число больше 0).

Установка атрибута правила: открываем правило Rule #3, нажимаем «(show options)», нажимаем знак плюса напротив, в открывщемся окне

выбираем значение «salience» в поле «Attribute» и в появившиеся поле записываем 10, см. рис. 28.

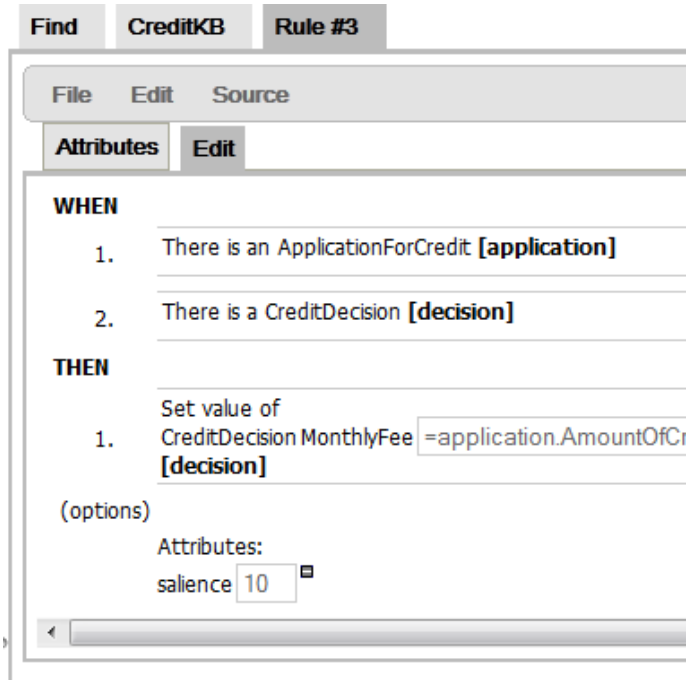

Рисунок 28 - Установка атрибута salience

Отлично! И последнее правило, которое мы создадим, будет следующим:

Если опыт работы < 1 года и
срок работы на последнем месте < 6 месяцев
То отказать в кредите.

Для описания этого правила, мы будем использовать ограничение «Any of (And)» из списка «Multiple field constraint», см. рис. 29.

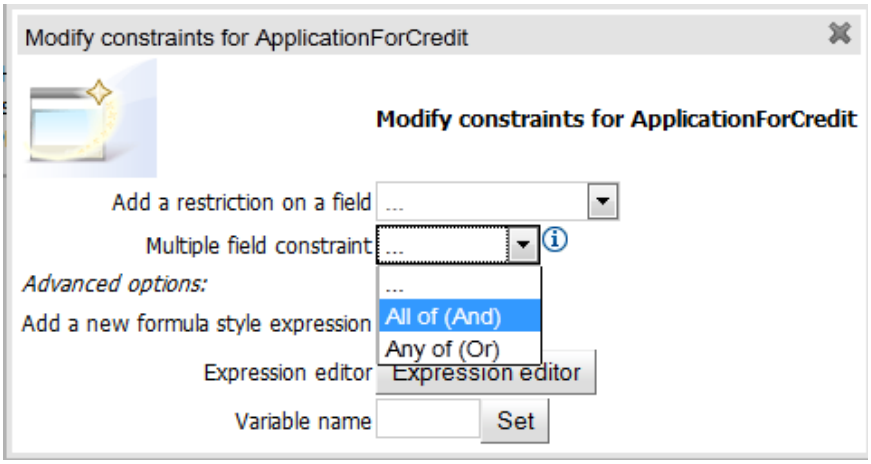

Рисунок 29 - Ограничение "Any of (And)"

Полностью завершенное правило Rule #5 см. на рис. 30.

На этом мы закончили с описание правил и готовы перейти к созданию тестовых сценариев, для проверки корректности работы базы знаний.

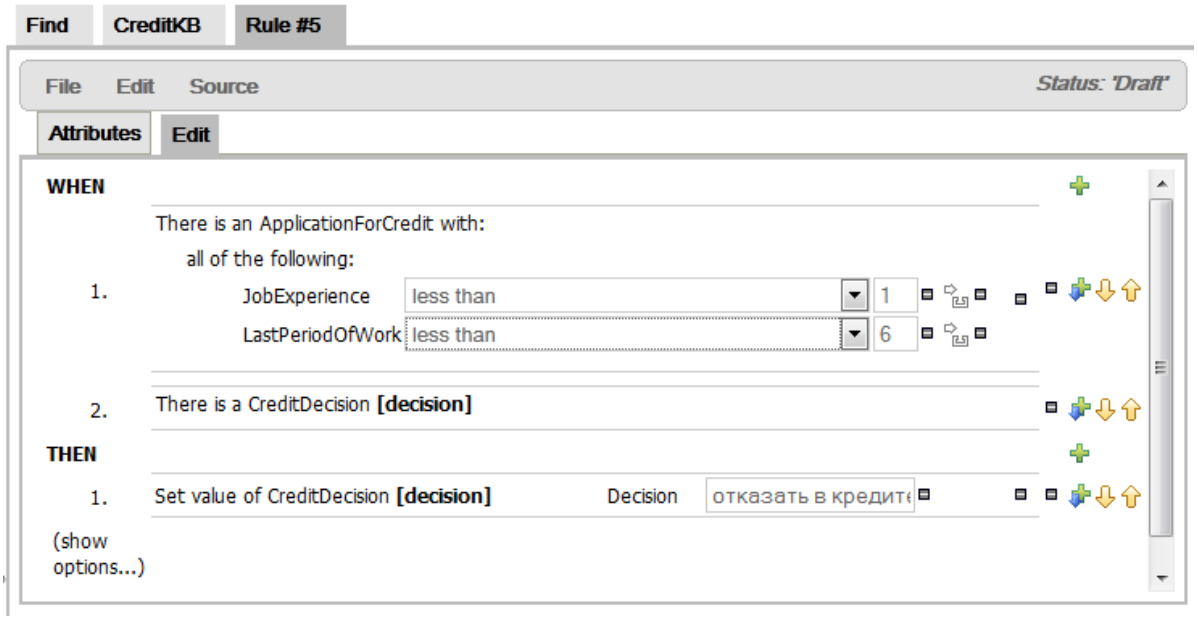

Рисунок 30 - Правило Rule #5

Лабораторная работа № 5. Создание тестового сценария

Приступим к покрытию нашей базы знаний тестами, т.е. написанию тестовых сценариев для тестирования логики базы знаний. С помощью этих тестовых сценариев мы должны проверить как можно большую часть логики БЗ.

Тестовый сценарий создаем так же как и правило, но в меню «Create New» необходимо выбрать подменю «New Test Scenario», после того как вы выбрали имя тестового сценария, для удобства в этом случае назовем его Test #1, откроется новая вкладка с именем теста, см. рис. 31.

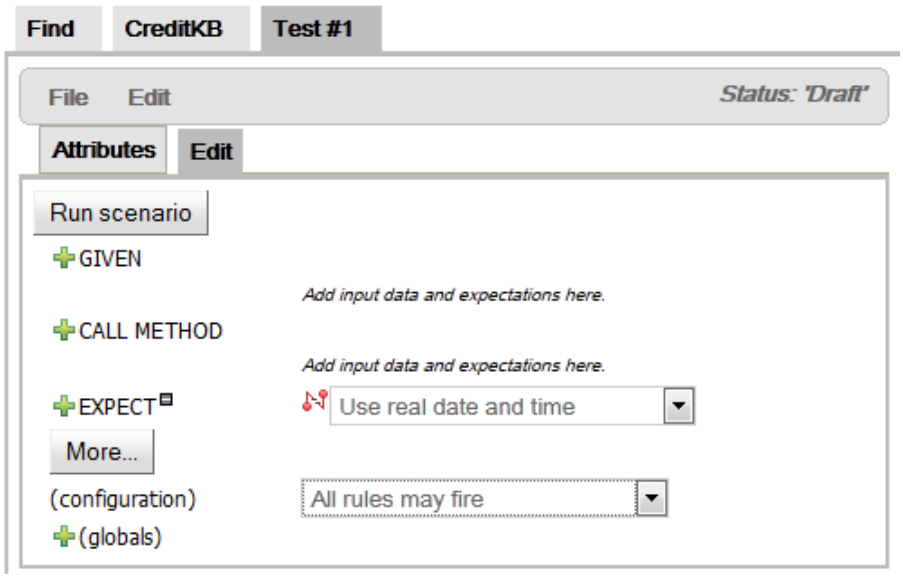

Рисунок 31 - Пустой тестовый сценарий Test #1

Тестовый сценарий состоит из двух основных частей: входные факты (часть «GIVEN») и ожидаемые факты (часть «EXPECT»).

Общий алгоритм работы тестового сценария следующий:

1. входные факты добавляются в рабочую память машины вывода,
2. запускает выполнение правил из базы знаний,
3. выполняется проверка наличия ожидаемых фактов в рабочей памяти.

Один тестовый сценарий может содержать несколько пар «входные факты – ожидаемые факты», чтобы добавить новую пару достаточно нажать на кнопку «More…».

Добавление входного факта

Начнем мы с простого теста, который проверяет работу правила Rule #1, для этого в качестве входных фактов мы добавляем факт типа ApplicationForCredit и факт типа CreditDecision. При добавлении входных фактов мы должны указать имя факта, назовем их «application» и «decision» соответственно.

Добавление входного факта: нажимаем знак плюса рядом со словом GIVEN, выбираем тип факт из списка «Insert a new fact», придумываем ему имя в поле «Fact name» и нажимаем «Add», см. рис. 32.

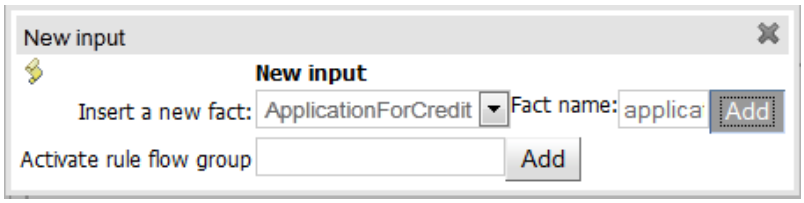

Рисунок 32 - Добавление входного факта

Выполним те же действия и для факта типа CreditDecision, тогда получим два входных факта в тесте Test #1, см. рис. 33.

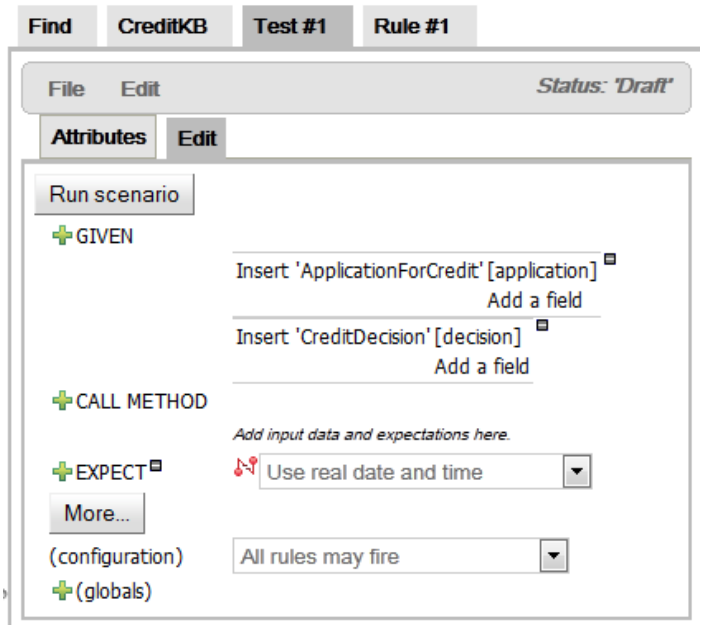

Рисунок 33 - Тест Test #1 и входные факты

Так как мы определили, что поле Decision факта CreditDecision должно иметь значение по умолчанию - «выдать кредит», то необходимо это учесть при разработке тестовых сценариев.

Установка значения полю входного факта: нажимаем «Add a field», выбираем необходимо поле из списка «Choose a field to add» и нажимаем «OK», результат см. на рис. 34.

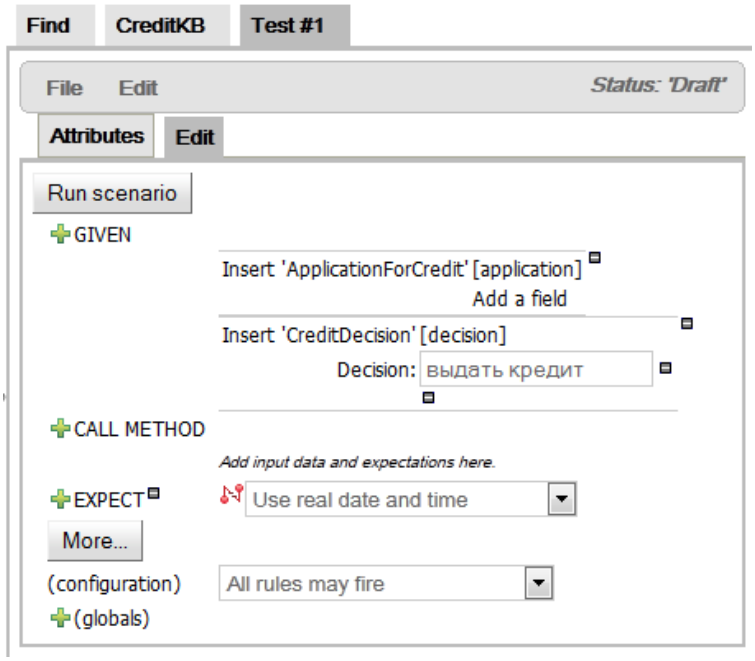

Рисунок 34 - Установка значения полю входного факта

Аналогично заполняем следующие поля факта ApplicationForCredit:

- AmountOfCredit: 100000,
- PeriodOfCredit: 24,
- Salary: 15000,
- Age: 17,
- JobExperience: 1,
- LastPeriodOfWork: 12,
- CurrentObligations: 0,
- Sex: M.

Теперь осталось добавить ожидаемый факт. В результате, мы ожидаем, что в поле Decision факта CreditDecision будет записано «отказать в кредите».

Добавление ожидаемого факта

Добавление входного факта как ожидаемого: нажимаем знак плюса рядом со словом EXPECT, в открывшемся окне из списка «Fact value» выбираем имя переменной, которой мы присвоили входной факт, и нажимаем «Add».

Теперь факт CreditDecision записанный в переменную «decision» является как входным, так и ожидаемым фактом. Теперь записываем «отказать в кредите» в соответствующее поле ожидаемого факта, готовый тестовый сценарий см. на рис. 35.

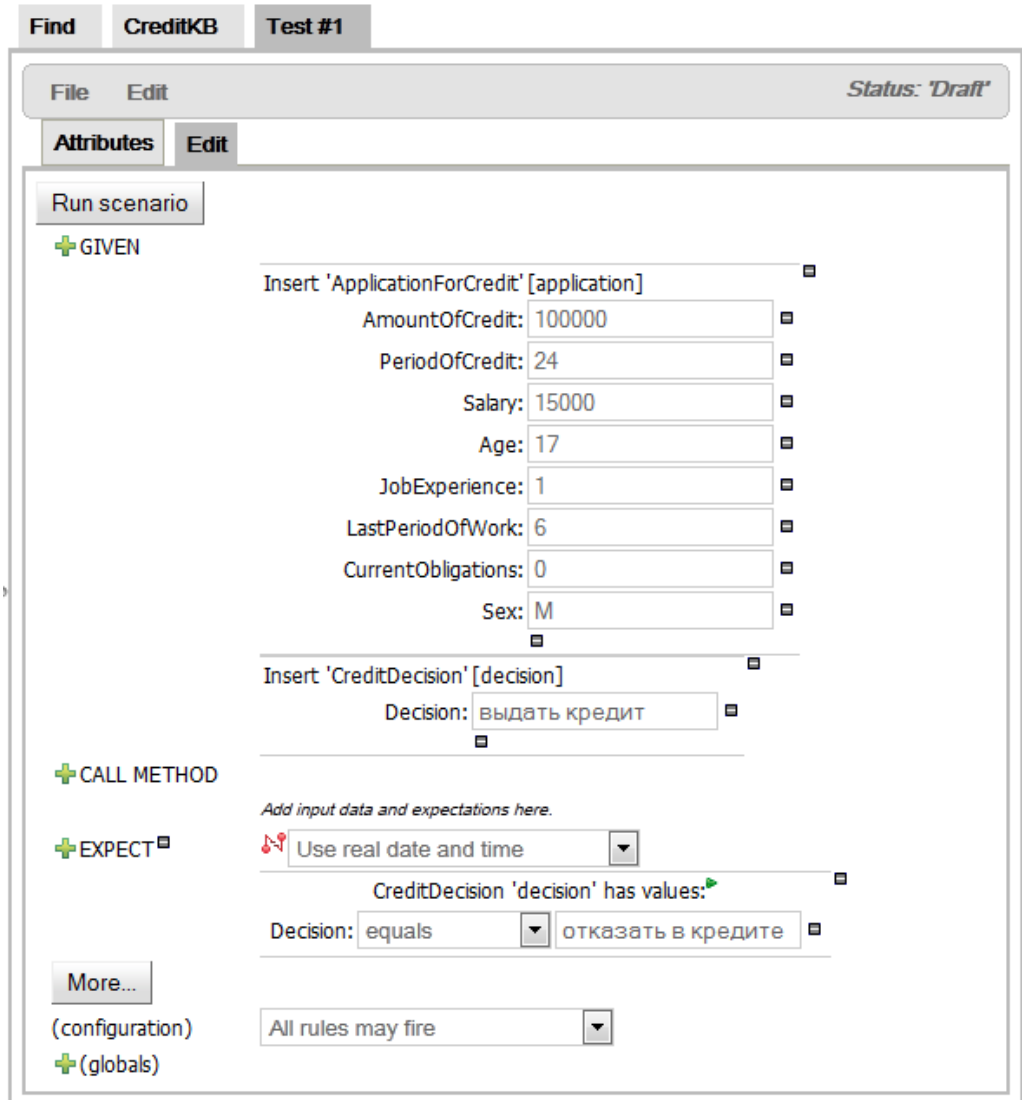

Рисунок 35 - Тестовый сценарий Test #1

После того как вы закончили реализовывать тестовый сценарий не забудьте его сохранить.

Запуск тестового сценария

Для запуска тестового сценария достаточно нажать на кнопку «Run scenario» и если тест прошел успешно, то вы увидите под этой кнопкой зеленый прямоугольник со 100%, как на рис. 36.

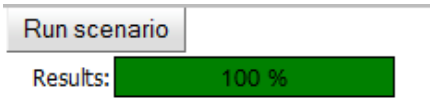

Рисунок 36 - Результат запуска тестового сценария

Так же для отладки теста будет удобно ознакомиться с отчетом, в котором перечислены все события, такие как добавление, удаление и изменение факта и срабатывание правил. Для просмотра отчет нажмите «Show events», см. рис. 37.

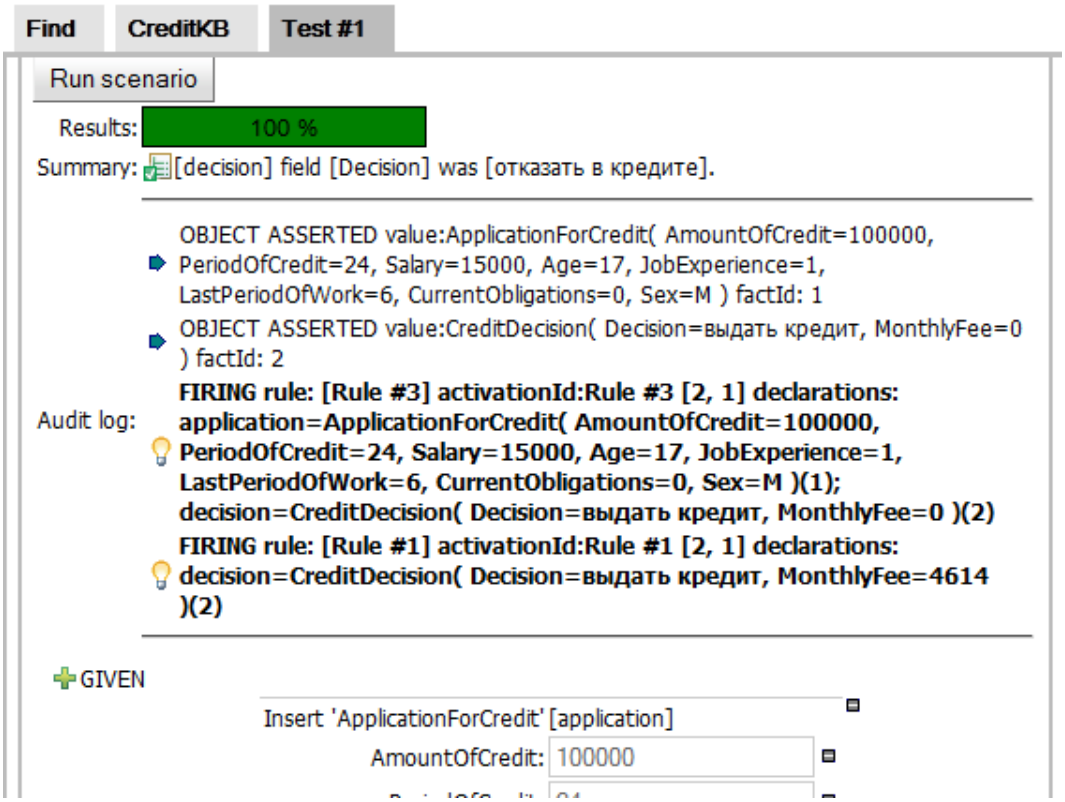

Рисунок 37 - Отчет по запуску тестового сценария Test #

Лабораторная работа № 6. Создание перечисления

В этой лабораторной работе мы займемся улучшением разработанной базы знаний, а именно мы добавим перечисления для полей «ApplicationForCredit.Decision» и «CreditDecision.Desicion».

Перечисление – это список возможных значений, которые может иметь поле факта. В редакторе правил для полей фактов, для которых определены перечисления, текстовое поле для ввода значения заменяется на выпадающий список.

Создание перечисления

Перечисления создаются, так же как и остальные сущности системы Drools, для этого необходимо в меню «Create New» выбрать подменю «New Enumeration».

Для описания перечислений используются три поля, каждая строка описывает одно перечисление:

- Fact – имя факта, в котором находится поле,
- Field – поле, для которого создается перечисление,
- Context – описание перечисления.

Опишем перечисление для поля «ApplicationForCredit.Sex», которое принимает два возможных значения ‘М’ или ‘Ж’. Для этого в поле «Context» нужно записать следующее, см. рис. 38.

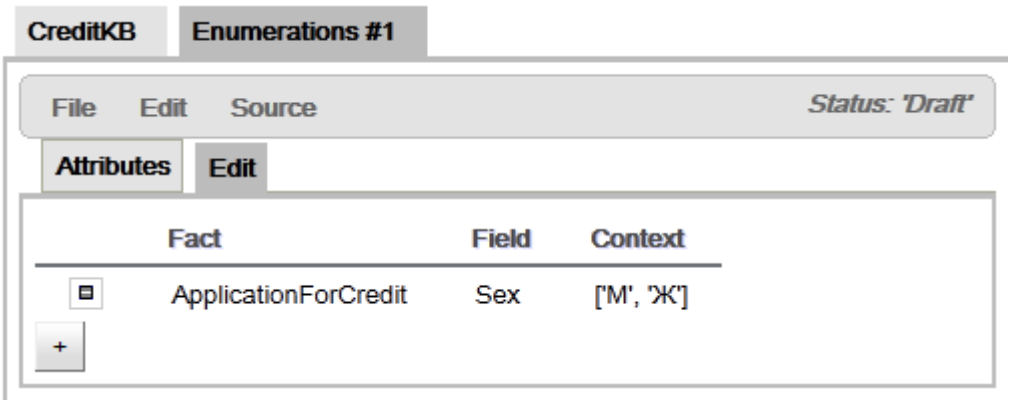

Рисунок 38 - Перечисления Enumeration #1

После того как перечисление создано, необходимо его сохранить и в подвкладке «Edit» вкладки пакета поочередно нажать на «Validate configuration» и «Build package», в результате чего система проведет валидацию всех сущностей пакета и сборку самого пакета.

Дабы проверить работу перечисления откройте правило Rule #2 и убедитесь, что теперь для поля Sex появилось выпадающий список возможных значений.

Более того созданное перечисление можно модифицировать таким образом чтобы для наглядности в редакторе правил вместо одиночных букв ‘М’ и ‘Ж’ отображались слова ‘Мужской’ и ‘Женский’ соответственно, а одиночные буквы использовались системой только для вычислений. Для этого в поле «Context» необходимо записать следующее:

[‘М=Мужской’, ‘Ж=Женский’]

После этих изменений правило Rule #2 примет следующий вид, см. рис. 39.

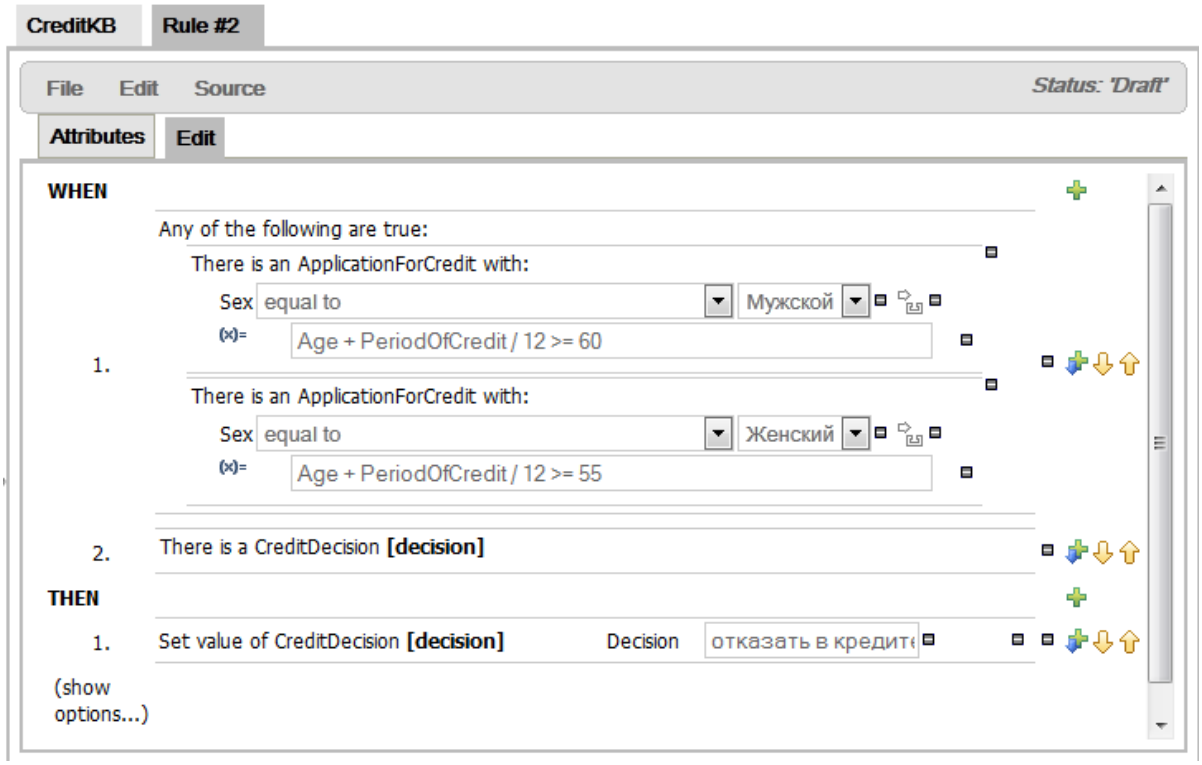

Рисунок 39 - Правило Rule #2 после изменения перечисления

Хорошо! Теперь вы научились создавать перечисления для полей факта. Чтобы закрепить полученные знания создайте перечисление для поля «CreditDecision.Decision» с двумя возможными значениями: «отказать в кредите» и «выдать кредит».

Заключение

В этом учебном пособии рассмотрены основные вопросы построения экспертных систем, включая вопросы жизненного цикла экспертной системы, стадии проектирования, коллектива разработчиков, а также теоретические вопросы работы продукционной системы.

Также в пособии содержатся методические указания к выполнению лабораторных работ с описанием основных возможностей системы управления бизнес-правилами Drools Guvnor. О расширенных возможностях системы вы можете ознакомится в документации пользователя[2] и в книге JBoss Drools Business Rules[3].

[2] Документация пользователя Drools Guvnor - http://docs.jboss.org/drools/release/5.5.0.Final/drools-guvnor-docs/html_single/index.html

[3] JBoss Drools Business Rules - http://www.packtpub.com/jboss-drools-business-rules/book

В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России, которым присвоена категория «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена программа его развития на 2009–2018 годы. В 2011 году Университет получил наименование «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

КАФЕДРА ПРОЕКТИРОВАНИЯ И БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ

ИСТОРИЯ КАФЕДРЫ

1945-1966 РЛПУ (кафедра радиолокационных приборов и устройств). Решением Советского правительства в августе 1945 г. в ЛИТМО был открыт факультет электроприборостроения. Приказом по институту от 17 сентября 1945 г. на этом факультете была организована кафедра радиолокационных приборов и устройств, которая стала готовить инженеров, специализирующихся в новых направлениях радиоэлектронной техники, таких как радиолокация, радиоуправление, теленаведение и др. Организатором и первым заведующим кафедрой был д.т.н., профессор С. И. Зилитинкевич (до 1951 г.). Выпускникам кафедры присваивалась квалификация инженер-радиомеханик, а с 1956 г. – радиоинженер (специальность 0705).

В разные годы кафедрой заведовали доцент Б.С. Мишин, доцент И.П. Захаров, доцент А.Н. Иванов.

1966–1970 КиПРЭА (кафедра конструирования и производства радиоэлектронной аппаратуры). Каждый учебный план специальности 0705 коренным образом отличался от предыдущих планов радиотехнической специальности своей четко выраженной конструкторско-технологической направленностью. Оканчивающим институт по этой специальности присваивалась квалификация инженер-конструктор-технолог РЭА.

Заведовал кафедрой доцент А.Н. Иванов.

1970–1988 КиПЭВА (кафедра конструирования и производства электронной вычислительной аппаратуры). Бурное развитие электронной вычислительной техники и внедрение ее во все отрасли народного хозяйства потребовали от отечественной радиоэлектронной промышленности решения новых ответственных задач. Кафедра стала готовить инженеров по специальности 0648. Подготовка проводилась по двум направлениям – автоматизация конструирования ЭВА и технология микроэлектронных устройств ЭВА.

Заведовали кафедрой: д.т.н., проф. В.В. Новиков (до 1976 г.), затем проф. Г.А. Петухов.

1988–1997 МАИ (кафедра микроэлектроники и автоматизации проектирования). Кафедра выпускала инженеров-конструкторов-технологов по микроэлектронике и автоматизации проектирования вычислительных средств (специальность 2205). Выпускники этой кафедры имеют хорошую технологическую подготовку и успешно работают как в производстве полупроводниковых интегральных микросхем, так и при их проектировании, используя современные методы автоматизации проектирования. Инженеры специальности 2205 требуются микроэлектронной промышленности и предприятиям-разработчикам вычислительных систем.

Кафедрой с 1988 г. по 1992 г. руководил проф. С.А. Арустамов, затем снова проф. Г.А. Петухов.

С **1997 ПКС** (кафедра проектирования компьютерных систем). Кафедра выпускает инженеров по специальности 210202 «Проектирование и технология электронно-вычислительных средств». Область профессиональной деятельности выпускников включает в себя проектирование, конструирование и технологию электронных средств, отвечающих целям их функционирования, требованиям надежности, проекта и условиям эксплуатации. Кроме того, кафедра готовит специалистов по защите информации, специальность 090104 «Комплексная защита объектов информатизации». Объектами профессиональной деятельности специалиста по защите информации являются методы, средства и системы обеспечения защиты информации на объектах информатизации.

С 1996 г. кафедрой заведует д.т.н., профессор Ю.А. Гатчин.

За время своего существования кафедра выпустила 4364 инженеров. На кафедре защищено 65 кандидатских и 7 докторских диссертаций.

Муромцев Дмитрий Ильич, Колчин Максим Александрович

Разработка экспертных систем в Drools Guvnor

Учебно-методическое пособие

В авторской редакции
Редакционно-издательский отдел НИУ ИТМО
Зав. РИО Н.Ф. Гусарова
Лицензия ИД № 00408 от 05.11.99
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Санкт-Петербургского национального
исследовательского университета
информационных технологий, механики
и оптики
197101, Санкт-Петербург, Кронверкский пр., 49